University of Twente
Enschede - The Netherlands

Design and Prototyping of a Human-Computer Interface for a Desktop Tele-classroom Conference Application

Author:	Marcel van Bergen
Date:	August 30, 2007
University:	University of Twente, Netherlands
Department	Electrical Engineering, Mathematics & Computer Science Architectures and Services of Network Applications
Graduation Committee:	dr.ir. I.A. Widya dr.ir. B.J.F. van Beijnum dr. ir. D. Konstantas

Table of contents

1 Problem definition

1.1 Summary

This chapter gives a general introduction and some definitions and explanations of the most important terms which will be used throughout this report. It also gives the main problem definitions and research questions that are relevant to the assignment described in this report.

1.2 Background

1.2.1 General

In 1997, the application protocols subgroup from the Tele-Informatics and Open Systems (TIOS) research group from the faculty of Computer Science of the University of Twente was doing research on network support for distributed education. This is done in co-operation with the research group Instrumentational Technology from the Faculty of Educational Science and Technology of the University of Twente. One part of this network support consists of a Desktop Tele-classroom Conference (DTC) system. Teachers can use such a system to give lectures to students, using workstations in a network environment instead of a classroom.

A first design and prototype of such a DTC system was already made in that time [1]. The DTC system discussed in this report is its successor. Part of this new system has already been made [2]. This report describes the design and implementation of a Human-Computer Interface (HCI) part for this system. This HCI -part will make it possible for end-users to use this system.

Due to circumstances, this assignment was suspended in 1998 although a lot of work was already done and a working prototype of the above HCI was already made. Now, in 2007, this assignment is allowed to be continued and to be finished. Since a lot of time has passed while the IT has developed enormously, an extra design cycle, using modern knowledge and techniques, is required to actualize this assignment. This assignment is now being done within the research group ASNA (Architecture and Services of Network Applications) from the faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS) of the University of Twente.

The next sections explain the two important terms, HCI and DTC, more precisely.
After that, a more detailed description of the background of the original assignment is given, along with a number of requirements.

1.2.2 Human Computer Interface

A HCI of a system defines the whole of all possible interaction between the human and the computer of that system. In this case, "system" refers to the DTC system that is being designed.

A HCI, therefore, incorporates more that just the layout and design of the graphical windows, buttons, menus, etc. To design a HCI, you first have to know what kind of actions both humans and computers can or must take and how they are related. This will be explained in chapter 3.

1.2.3 The Desktop Tele-classroom Conference system

The objective of a DTC system is to be a system that can be used by a teacher to lecture students that are geographically separated, instead of using a real classroom. This means that it must at least support the functionality of a real classroom for its users. The DTC system therefore acts in fact electronically like a "virtual classroom" using a number of computers in a network that can exchange all kinds of information like audio, video and text that is necessary for teaching. In the future, the quality of a lecture can be improved by providing extra tools that can only be used in such an electronic classroom. The DTC system will be further explained in the next paragraphs.

1.2.4 Lecture

In the remainder of this report, the term lecture will be frequently used. Within the scope of this report, "lecture" is meant to be a lecture in the classical form, where one teacher, presents his course to a group of students.

1.3 More detailed background and first requirements

1.3.1 Use of different classroom descriptions

For this prototype, the emphasis is on providing a substitute, virtual, classroom, as mentioned in Section 1.2.3. To know the substitute, the best way is first look at the original. We therefore first have to analyse what happens during a lecture in a real, face-to-face, classroom.

To do this analysis, first a description of a real, face-to-face classroom situation is needed. This is given by sketching the most important situations during a lecture in such a classroom. Then a number of requirements are given for the virtual classroom, which are necessary to substitute such a real classroom. That is followed by the description of the most important situations during a lecture in a virtual classroom using a DTC system. These will be used and extended later in this report.

1.3.2 Description of a real, face-to-face classroom

Figure 1

A real, face-to-face classroom is a room in which both students and teacher are physically together to exchange audio, visual and textual information in order for a teacher to give lectures to students.

Most important situations during a lecture in an ideal version of a "real" classroom:

1. When a teacher in a real classroom gives a lecture according to the definition above, he takes part in and controls most communication. He presents his lecture by talking, presenting sheets, drawing on a blackboard, making gestures, etc.

2. Occasionally, a student who wants to ask a question, raises his hand, waits for permission to speak and then poses his question in the class. The teacher then answers the question in the class.

3. The other way around, a teacher sometimes asks a question to his students and waits for someone give an answer or chooses a specific student who has to give the answer.

4. When the teacher requests silence, immediately all students stop speaking and listen to the teacher.

1.3.3 Requirements for a DTC system

Within a DTC system, there is no face-to-face contact between the different users so there is no means of physical exchange of information. All exchange of information must be done electronically.

To give a lecture using such a system, there are a number of minimal requirements for information exchange:

- When we look at part 1 within the section above (Section 1.3.2), we see that the teacher must at least have facilities to transmit audio, so students can hear him. Also a means to visualise sheets and drawings, transmission of video, text or electronic drawings, is needed.

- When we look at part 2 and 3 within the section above (Section 1.3.2), we see that the Students must have the means to pose or answer questions. That means that they must at least have the possibility to transmit text or audio, so the teacher and other students can hear or read the question.

- When we look at part 2, 3 and especially 4 within the section above (Section 1.3.2), we see that the teacher must have the means to control the lecture.
  To control a lecture, the teacher in a real classroom is dependant on "social protocol" For instance, a student is not supposed to interrupt the teacher at random moments. He risks a reprimand if he does so. He could eventually even risk removal from the classroom.
  This social protocol, on its turn, is dependent on more than just audio-information. For instance a student raises a hand to ask permission to speak. Within the DTC system, there is not always video information available from students. Also, the "psychological distance" between participants is larger because each is sitting behind his own computer in his own physical environment. Because of this we will not rely on the social protocol alone, but have chosen for a floor control mechanism. This is a mechanism that substitutes amongst others "hand raising" and enables the controller, i.e. the teacher, to permit or refuse exchange of information like audio or video from all others on an individual base.

Hereafter, the DTC system and a lecture situation within such a DTC system will be described using the sketched situations within a real classroom and the above requirements.

1.3.4 Description of a DTC system

A DTC system provides a "virtual classroom". It does this by using a number of computers in a network that can exchange audio, video, image and text information in order for a teacher to give lectures to students without physically being together (see Figure 2).

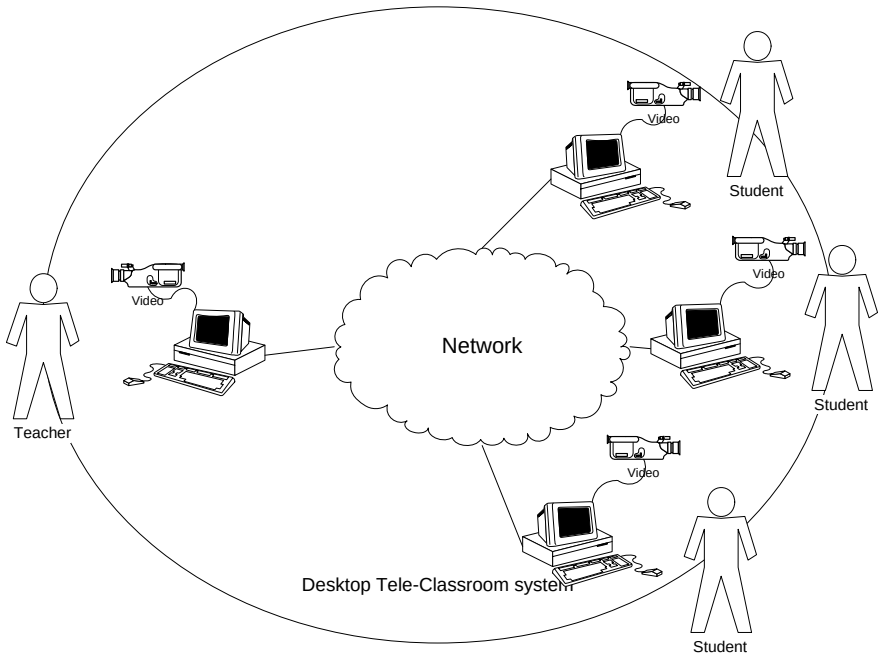


Figure 2

Description of the most important situations during a lecture in a Desktop Tele-classroom Conference system:

The numbering, used below, corresponds to analogue situations as given in Section 1.3.2 for a real classroom.

1. Like the lecture as described earlier in a real classroom, the teacher within this system takes part in the communication and has all the control. He uses the DTC system to present his lecture.

2. When a student wants to pose a question (In order to do that, he must be able to transmit audio, text and/or video), the student first has to request explicit permission of the teacher. When that permission is granted, everyone can hear / see his question and the teacher's answer.

3. When the teacher wants to ask a question to his students, he can ask the question and wait for someone to request permission to give an answer. Alternatively, he can choose a specific student who then has to give an answer immediately.

4. Since the teacher has all the control, the transmission-rights of a student can be taken by the teacher at any time, to interrupt a student.

1.3.5 Differences with existing teleconference systems

Typical, already existing teleconference systems, like Cu-Seeme, IVS and MMCC [1] or Showme [Sun Microsystems], provide their users also with means to exchange data, like audio, video text, images and drawings. In some cases a teleconference system provides means for a number of persons to enforce control over this exchange to give the conference more structure [4].

The main difference between these already existing teleconference systems and the DTC system reported here is that this system behaves towards the users as a virtual classroom and not as a general conference-tool. Furthermore, there is more centrally enforced control over all data-exchange than in a normal conference system.

The DTC system is designed to eventually use ATM networks instead of ISDN, Ethernet or Internet. Therefore the number and quality of audio and video-connections can eventually be much larger than for the older systems like CU-Seeme.

1.4 Goal

The goal of this assignment was to design and implement a Human Computer Interface (HCI) for a DTC system that provides all necessary means to give a lecture as defined in Section 1.2.4.

1.5 Research Questions

The design and implementation of this DTC HCI can be guided by a number of problems/questions that have to be solved.

- First of all, a design-method for HCI's has to be found.
- The requirements and functionality of a Desktop Tele-classroom Conference has to be defined exactly.
- Design- and implementation-constraints have to be found. These are for example imposed by the available hardware or other research and implementations within our research group.

1.6 Organisation of this report

This report exists basically out of 2 parts. The first part, consisting of Chapter 0 to Chapter 7, treats the initial assignment. The second part, consisting of Chapter 8 to Chapter 13, treats the actualization of this assignment: an extra design cycle for the HCI of the DTC system.

After this initial chapter, first all initially used literature will be treated in Chapter 2. At the end of this chapter, the known constraints for the design and implementation of the HCI for the DTC system are given.

In Chapter 3, the information gathered in Chapter 2, together with the already given requirements in Chapter 0, will be used to derive all user requirements and a design-approach for the HCI of the Desktop Tele-classroom Conference system.

The actual design is given in Chapter 4.
This design is derived from the user requirements and approach as given in Chapter 3, together with the design-oriented constraints that are given in Chapter 2.

The design from Chapter 4, together with all implementation-oriented constraints from Chapter 2, result in an implementation of the HCI for the DTC system that will be described in detail in Chapter 5.

In Chapter 6, the implementation as described in chapter 6 will be evaluated.

Chapter 7 contains the conclusion and some recommendations for the old DTC system from the initial assignment.

Chapter 8, introduces the actualization of this assignment.

In Chapter 9 a new literature study will be introduced to actualize this assignment. At the end of that chapter, new constraints for the design of an improved DTC system, with emphasis on the HCI of such a system, using current knowledge and technologies will be given.

In Chapter 10 the information from Chapter 9 will be combined with the existing user requirements from Chapter 3 to arrive at new user requirements for such an improved DTC system.

In Chapter 11, the above user requirements, together with the new constraints as given in Chapter 9, will be used to give a number of recommendations for a new HCI design. Since this is done as an actualisation of an assignment, it will not result in a complete design. Also a number of recommendations will be given for improvement of a selected current teleconference system.

Chapter 12 is an elaboration on a few design and implementation details. It will not result in a design for a full working prototype of a new DTC, but will allow the exploration in some detail of some modern techniques and theories.

Chapter 13 contains the final conclusion and recommendations of this report.

2 Literature study

2.1 Summary

In this chapter, various literature will be examined. First, a number of topics of relevance will be established. Then, the ways of finding literature will be discussed. After that, all found literature will be presented, along with its respective contribution with respect to the earlier found topics of relevance. As a conclusion, a number of design and implementation constraints, mainly as a result of this literature examination, will be presented.

2.2 What is relevant?

To find and evaluate literature, first a number of topics of relevance must be established. These topics must be relevant to the designing and or implementation of the Human Computer Interface (HCI) of the Desktop Tele-classroom Conference (DTC) system.

These topics can be ordered in two separate groups:

Design

- First, it is important to get a number of examples of the development of other HCI's. These give examples of useful design methods, provide adequate terminology to describe all aspects of a HCI design, and give hints to additional methods to solve small practical design problems.
- Using these examples, a suitable design method has to be found that can be used to make the actual design of the HCI of the DTC system.

Implementation

- Additional design constraints for the Human Computer Interface of the Desktop Tele-classroom Conference system must be found, if existent. These are not only determined by available hard- and software, but also by related projects within our research group and by existing standards.
- In a later stage of the design or during implementation, it is very useful to have a number of practical design tips for HCI's.
- For the actual implementation, a suitable development environment has to be found. This can consist of one or more computer-languages and, maybe, additional graphic design-tools. These can't always be tested in advance, so descriptions of such environments are important.

2.3 Where to search?

There are a number of places to look for relevant literature. Some are specific for the topic of relevance and will be described below.

To find a general, standardised design method, the best place to look is in ISO documents.

For additional design constraints, the best place to look would be in reports, made by members of our own department (e.g. to find descriptions of already implemented parts of the DTC system and or to look for possibilities of co-operation between this DTC and other applications).

Practical design tips can best be found in books or lecture notes since they are mostly a compilation of a number of different areas of research.

Descriptions of programming / development environments can be found mostly in online-documentation and in specialised books that each handles one particular programming environment.

2.4 Examined literature

2.4.1 Finding a design method for HCI's

As described in Section 2.2, a design method for HCI's has to be found. This can be done, amongst others, by looking at examples of HCI-design.
The paper about "Project Platinum" [4] describes a complete design of a HCI and illustrates the whole design process of a HCI for a conference system. It contains some ideas about floor control and a generic teleconference model. This paper refers to a standard design method that was developed by ISO for the design of HCI's.
The design-method mentioned above is given in a standard ISO document, "guidance on Usability" [3]. This is a paper about a standardised design and test method for HCI's based on usability and contains besides a design method some suggestions about how to measure usability.
These two papers provide us with a standard design method for the HCI for the Desktop Tele-classroom Conference system, together with a number of examples of the usage of this method.

2.4.2 Additional constraints

To get additional design and implementation constraints, a number of reports, made by members of our own department, as suggested in Section 2.3, were studied.

The first report was about the first design and implementation of a DTC system by Vincent Korenromp [1] that initiated this project. This report can't contribute much to this part of the project anymore.
A second report is about the implementation of a session-layer that will be used for a DTC system that is capable of transport of audio, video, text and protocol-information by Alex Jongman [2]. This implementation was build for Sun workstations equipped with Sun video-digitising hardware and optional with ATM network cards. From this report it is clear that the, by the session layer provided service to implement floor control, is mostly a transport-service and not enough to use for floor control. Most of it has to be implemented in the layer above this session layer.
The last report is from Xanter Wilhelm [5] and treats a design of the logistic support for a larger tele-education environment. The DTC system could be made in such a way as to make it easier to integrate user-data, provided by this logistic support system.

To have an idea of existing standards for teleconferencing, as suggested in Section 2.2, a document, from ITU, called "Real time audio-visual control for multimedia conferencing" [12] was read. This document defines a standard architecture with standard functionality for teleconference systems. The borders between different modules and layers as already laid out by Alex [2] have a lot of similarity with the described architecture in this ITU-document.

To get some practical guidelines on HCI implementation, as suggested in Section 2.3, lecture notes made by Gary Perlman, [6] were read. These lecture-notes contain numerous practical design and implementation tips.

As suggested in Section 2.3, an evaluation was made of a number of languages and environments that are capable of generating a program within the desired hard- and software environment. Some attention has been given to cross-platform development, possibilities to implement an extended graphical user-interface and the ability to adapt existing code later.
The final recommendation was to use the language TCL/TK, developed by SUN Microsystems. This language can be used on multiple platforms and has some good public-domain development-tools available.

In Appendix A, all the literature described below is presented more comprehensively.

In Appendix B, the search for a programming and development environment is presented in detail.

The found design- and implementation-constraints from this chapter and from chapter 0 are presented short in Section, 2.5.

2.4.3 Results

A table that summarises and enhances the results from this section is presented in Table 1.

Search results of literature study

Article	General	design/ implementation constraints	Design method	Examples	tele conf. Ideas	Tips
ISO DIS 9241-11, *ergonomic requirements for office work with Visual Display Terminals: Part 11 guidance on Usability* **[3]**	Paper about a standardised design and test method for HCI's based on usability.	-	stan-dardised method	One small example of design.	-	a table with a number of suggestions about how to measure usability
Project Platinum, Conference Management Application Design **[4]**	This paper describes a complete design of a HCI.	-	a few help-methods (using roles, hci-views)	illustrates the whole design process of a HCI	floor control, using general teleconfere nce system	-
V. Korenromp, *On the prototyping of a Multimedia Conferencing System for Desktop Tele-classroom Conference Application* **[1]**	Is initiator: Report about the first design and implementation of a DTC system.	-	-	Gives descriptio ns of some existing teleconfer encing systems	-	-
Alex Jongman, *On the Prototyping of Audio and Floor Control Protocols in a Desktop Tele-classroom Conference Application.* **[2]**	Report about implementation of a session-layer for a DTC system that is capable of transport of audio, video, text and protocol-information.	Implementation: interface, session-possibilities, Design: floor control	-	-	floor control	-
Xanter Wilhelm, *Logistics for the Off-line Communication in a Desktop Tele-education Environment.* **[5]**	Report about design of the logistic support for a larger tele-education environment	Implementation: possibility to connect education-environment to DTC	-	-	-	-
ITU, *Real time audio-visual control for multimedia conferencing.***[12]**	Defines a standard architecture with standard functionality for conference systems	Similarities with implemented session-layer [2] Defines conference model	-	-	-	-
Gary Perlman, *User Interface Development* **[6]**	These lecture-notes describe numerous practical design and implementation tips.	-	-	Extended examples of measure-ments	-	Many (>10) tips about implemen-tation and measuring
Appendix B	Search for a programming/ development environment.	Implementation constraints (use of TCL/TK)	-	-	-	-

Table 1

2.5 Constraints

2.5.1 Hardware constraints

For now, because of historic reasons ([1], [2]), the system will be developed using Xwindows/Solaris on a Sun Sparc-station. These systems can need to be equipped with Sun video-digitising hardware and, optionally, with ATM network cards.

In the future, this DTC system will probably be used on more than one type of computers. This must be taken into account.

2.5.2 Software constraints

To implement a real working Desktop Tele-classroom Conference system, only a HCI is not enough (see above). Therefore, this HCI will be built upon a session layer that was being designed and implemented by another student in this group [2]. This session-layer will take care of the actual connection establishment and of transport of all audio, video, text and control information. In addition, it will present audio and video directly to the users. The interface between this above-mentioned session-layer and the HCI-layer must be well defined in terms of data and functionality and must be tested. Any obscurity must be clarified. Any problem must be solved, if possible.

Because of the above hardware constraints, the HCI implementation must be made in such a way that it is possible to convert the HCI from one platform to another without much difficulty. This sets some serious demands on the used development languages and tools. This has been investigated in Appendix B. The final conclusion from that investigation is that the language Tcl/Tk in combination with some further development tools will best suit these demands.

There must be a way to simplify integration of user-data as provided by the logistic support system as described in [5] into the DTC system.

2.5.3 Functional constraints

From Chapter 0, we can deduce the following constraints: The teacher must at least have facilities to transmit audio, so students can hear him. In addition, means to visualise sheets and drawings, like transmission of video, text or electronic drawings, are needed. Students must have the means to pose or answer questions. That means they must at least have the possibility to transmit text or audio, so the teacher and other students can hear or read the question (see Section 1.3).

There must be a floor control mechanism. This is a mechanism that enables the teacher to permit or refuse transmission of information, like audio or video, from all others on an individual base (see Section 1.3.4). This is only partly implemented within the used session-layer [2] and therefore this must be designed and implemented as well.

The ITU document [12] defines a standard architecture and conference model. The architecture is already similar on certain points. Where it differs (non-separate transmission of audio/video and control information, no remote control of devices) it is caused by practical, technical, reasons.

3 User requirements

3.1 Introduction

Within this chapter, the user-requirements for a HCI for the desktop Tele-classroom Conference system will be determined.

The HCI for the Desktop Tele-classroom Conference system will not be designed directly from the outlines of the DTC system as sketched in Chapter 0. Instead, a more generic conference-model will be used that is suggested by the Platinum document [4]. This model provides better extendibility in a later stage.

Only the first design-steps will be made for the actual DTC system as described in Chapter 0 to provide both a mapping to the generic Conference system and a number of constraints for the actual implementation of a Generic Conference system that can be used as a DTC system.

The user-requirements will be derived from the known design constraints with the first steps of the method suggested by ISO 9241-11 [3] and the platinum document [4]. These documents describe a standard method to specify the usability of HCI's that uses actors, roles and scenarios to describe all interactions between humans and computers.

3.2 The 2-steps approach

The DTC system is based on an older design [1] that did not incorporate any real HCI considerations. The DTC system from this second design cycle will be the first to offer a complete HCI. In the future, within a next design cycle, the HCI-definitions might be extended to incorporate other educational schemes as those described in Chapter 1. Examples of such schemes are working classes, in which the students can communicate in small groups with each other or student-lectures in which a student-lecturer provide and control most of the lecture while the teacher only supervises. This means that in the future it might be necessary to introduce new kinds of users and functions. Alternatively, the needs and behaviour of existing users may become different from what we expect them to be at this stage.

It is therefore that we want to look for a more general description to incorporate a superset of the functionality of the desired DTC system. This will be done by using a generic Desktop Teleconference model that is suggested by the Platinum document [4]. The functionality of this generic Desktop Teleconference (DT) model will be used to further design and implement the actual Desktop Tele-classroom Conference system.

This DTC system will need a mapping from this generic DT model to the specific DTC system that was already outlined in Chapter 0. The method to make a mapping from the already outlined DTC system to a more generic DT model, is by making use of actors defined within one system that play roles within another system. This is also illustrated in the Platinum document [4].

In doing so, we have made a design that can incorporate changes more easily because, in most cases, only the mappings from actors within a specific DTC system to roles from the generic DT model have to be changed to create a new and different DTC system.

We can't make such a design for the HCI of a more general DTC system in one step. To make a good mapping possible from the already outlined specific DTC system to a more general DTC system, both sides must be known in more detail. The first design cycle will therefore be the direct initial HCI design for that specific DTC system as outlined in chapter 1. The next cycle will be the design of a more generic Desktop Tele Conference system, by giving a design according to a generic conference model, followed by a mapping from the specific DTC system to this more generic system to enable us to construct a more generic DTC system.

The actual implemented system will not be a complete generic conference-management system. Some functions will not be implemented yet. However, the architecture to implement and extend the system will be there.

3.3 DTC application oriented HCI-requirements

3.3.1 General description of the desktop Tele-classroom Conference (DTC) system

This section describes the general situation for the use of a DTC system. It is mostly derived from Section 1.3.4. This is combined together with reports about design and implementation of an earlier version [1] of this system and the session-layer for the present system [2].

1 In this DTC system there is one teacher, who gives a lecture to a number of students, who attend the lecture. The teacher presents his lecture using all available media (audio, text, video, etc.: this term will be explained in the next section). The teacher controls the whole DTC system and therefore also all permissions from all students for all available media.

   Students and teacher both have standard sink-permission for all media, but only the teacher has standard source-permission for all media.

2 When a student poses a question, he has to gain access to one ore more media. To do so, a student first has to request explicit permission of the teacher to access those media. When that permission is granted, everyone can hear / see that student's question on the granted media. The teacher's answer is received by all students on all media the teacher wishes to use.

   A student that has requested permissions for one ore more media can always withdraw that request.

   Once a student has got permissions for a medium, he can always give them up voluntarily.

3 When the teacher wants to ask a question to his students, he can ask the question and wait for someone to request media write permission to give an answer or he chooses a student who then is forced into accessing the selected media and who has to give the answer.

4 Since the teacher has all the control, the permissions of a student for any medium can be taken away by the teacher at any time, to interrupt that student.

3.3.2 Actors, roles and scenarios: a description

To make a good design according to the ISO [3] standards, all types of users have to be defined. These types are called "actors" (e.g., the actors of a football game are player, referee and coach). These types of users all can have one or more functions. These are called "roles". For instance, a player can be a lead, a defender, a keeper etc. For different global situations, types of users can have different functions. The total description of all functions in one global situation is called a "scenario" (you could, for instance, let a football team play volleyball: then you have the same actors, but with different functions).

The definitions are as follows:

An actor: This is a user of the system to be described with a certain number of roles to perform within that system. The name and functionality of an actor do not have to be the same as the real function in society. For instance: A referee might act as a player and vice-versa.

A role: This is a coherent set of distinguished functions within the system, performed by one actor. An actor can have more than one role.

A task: The description of one single function for an actor-role to be performed in one global situation within the system. This is mostly given by a short, simple description.

A scenario: The total description of all roles and their main tasks in one global situation within the system. This is mostly given by a short, simple description of the main tasks that can or must be performed for all actor-roles in one global situation within the system.

3.3.3 Actors, roles and scenario's in the desktop Tele-classroom Conference system

In the DTC system as described in the previous part, two actors can be distinguished: T and S. There is only one scenario, the lecture scenario. The actor-roles for this scenario are simply defined: There is one teacher who gives a lecture. The Students attend this lecture.

Actors and roles for the lecture-scenario:

Actor	Role	Function
T	Teacher	Providing a lecture to a number of students.
S	Student	Attending a lecture given by one teacher.

3.3.4 Detailed lecture scenario for the desktop Tele-classroom Conference

This is a more detailed description of the roles in the lecture scenario. The lecture scenario is, at the moment, the only scenario for the use of a desktop Tele-classroom Conference system. This description has been divided in a separate part for the role of the Teacher and the role of the Student.

Role description for Teacher:

> The teacher provides a lecture to a number of Students.
>
> A teacher enters the desktop Tele-classroom Conference and must identify himself. Optionally, he must be able to check for the presence of all subscribed students. After that, he gives the lecture, which consists of the lecture itself (audio + video), a presentation of lecture notes and the answering of questions by students and/or the posing of questions to students. The teacher has to do the accompanying management of local media and student floor-access rights. Managing floor-access rights means the teacher has to be able to view, acknowledge, or refuse requests for floor-access. When the lecture is finished, the teacher leaves the desktop Tele-classroom Conference.

Role description for Student:

> A student attends a lecture give by a teacher.
>
> A student enters the desktop Tele-classroom Conference and must identify himself. After that, he attends the lecture, which consists of the lecture itself (audio + video), a presentation of lecture notes and the posing of questions to the teacher and/or the answering of questions posed to him by the teacher. When the lecture is finished, the student leaves the desktop Tele-classroom Conference.

A point wise lecture scenario for both teacher and student can be found in appendix C.

3.3.5 Task analysis for the lecture scenario

The tasks analysis can be seen as a refinement of the scenario. Given the above lecture scenario, a number of goals and tasks can be identified for each actor and each role.
These are point wise presented in Appendix C.

The result of the scenario in Appendix C is a global description of the cohesion and functionality of all defined roles. The result of the task analysis in Appendix C is a large set of tasks that can be performed for each role within the given scenario.

3.3.6 Definition of "Floor"

The term "floor" has to be explained, since it will be used frequently in the next sections.

In the Platinum project [4] and in the report about the implementation of the session layer [2], an elaborate view about teleconferencing is presented. This view implies that a conference has not just audio and video connections, but has controlled media, called "floors". These floors are separate universal transfer-channels that can distribute audio, video, text, or whiteboard information for which both source- (i.e. read) and sink- (i.e. write) permissions are managed by a controller.

Each floor can be separately controlled for source/sink access of all users by its controller. The controller can view all pending requests and acknowledge or refuse them at will. Requests can be withdrawn at will by the requesting user. Permission to access a floor can be granted without a prior request.

Within a DTC system, the use of floors gives the teacher a good method of control over the lecture. These terms and ideas, like the use of separate, different, controlled media were therefore integrated in the description for a desktop Tele-classroom Conference system.

The above-mentioned session layer [2] that is being used, does not provide a complete service for floor control yet. Only a service is available to transport all necessary information to implement floor control. A working version of floor control, however, is necessary to design and implement an acceptable HCI for the DTC system. This means that floor control itself also has to be designed and implemented. Floor control will make use of the already available transport functionality of the session layer [2].

3.3.7 Entity-Relationship diagrams

To describe the actors, their roles, their mutual interaction and their interaction with the system itself more clearly, point wise scenario-definitions and task analysis will be used.

In addition to these, Entity-Relationship (E-R) diagrams will be used in this report. The reason for their use is that they give a more clear survey of the whole system. Another advantage of E-R diagrams is that they give an accurate definition of the coherence of all different parts of a system.

These E-R diagrams will be presented with a stepwise refinement to introduce details at a later stage. The last diagram contains a model of the whole system in full detail.

Figure 3 is a first E-R model of the DTC. It relates both types of actors to each other and to the DTC system. There is only one lecture and one teacher at a time within the confines of this system.

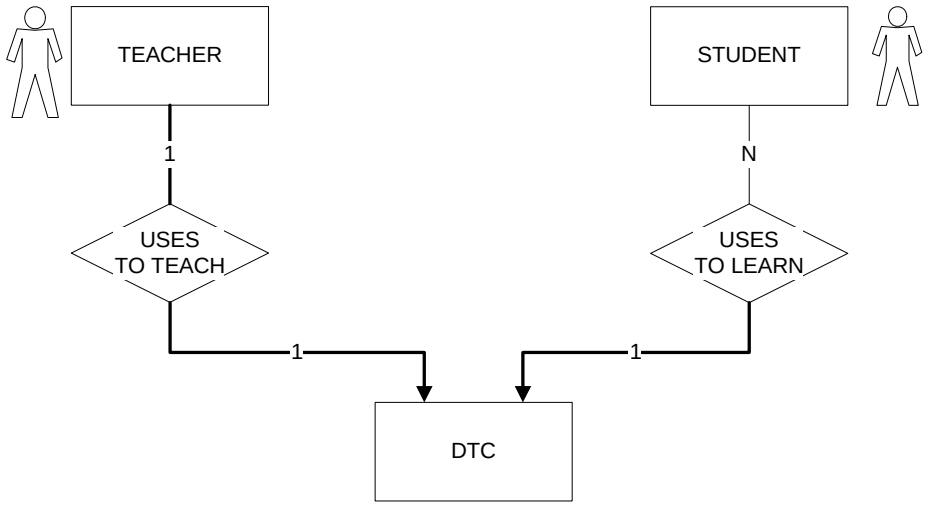


Figure 3: The actors and the system.

A Teacher uses a DTC to teach a number of students.
A student uses a DTC to learn something from a teacher.

Hereafter follow two more E-R diagrams that give a more detailed model of a Desktop Tele-classroom Conference system.

Figure 4 adds general control to the DTC system. The teacher controls the DTC. The student does not.

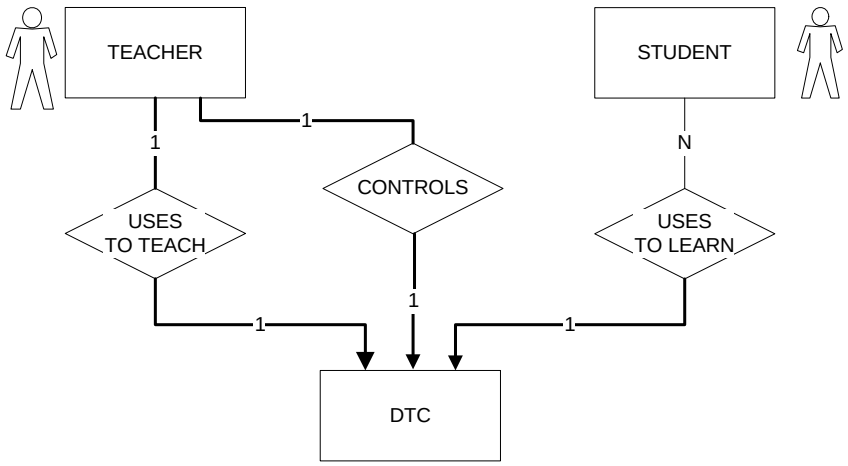


Figure 4: The teacher has total control over the DTC.

Figure 5 is the final diagram and adds media and media-control to the DTC system. Media and their control are further called floors, as described in Section 3.3.6. Floors are controlled by the teacher. Floors are accessed by both the teacher and the students, according to their rights.

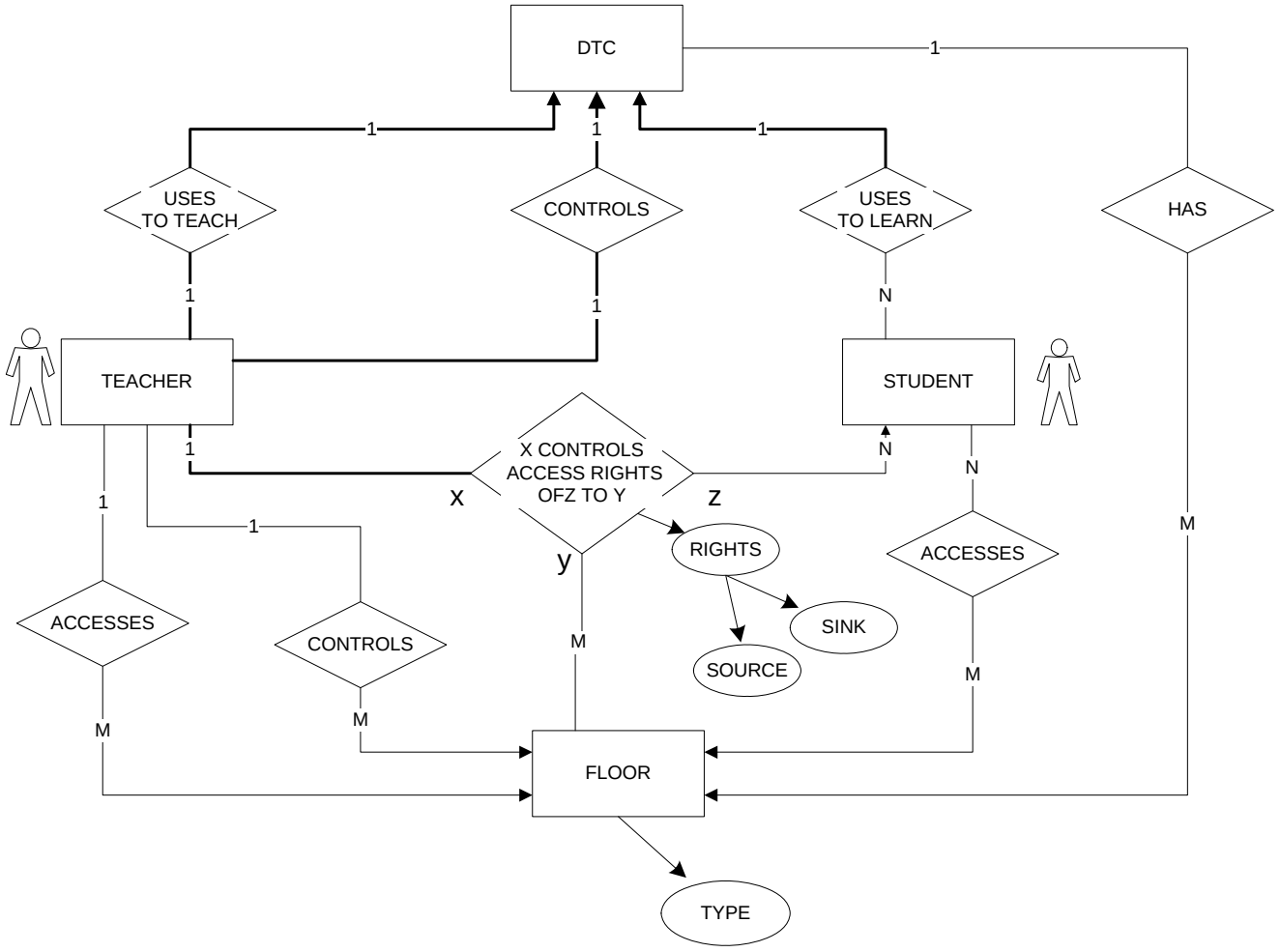


Figure 5: Complete diagram.

A teacher controls the DTC, all its different types of floors and the access-rights of all students to all floors.

3.4 Generic desktop teleconference model oriented HCI-requirements

3.4.1 A generic Desktop Teleconference model

A generic Desktop Teleconference (DT) model, according to the ideas presented in the platinum document [4], models a conference that is hierarchically organised in groupmeetings and floors. Floors were explained in Section 3.3.6 and form a more generic way to use different media like video or audio. Each conference, group and floor has a single controller and a number of participants.

A groupmeeting consists of a number of floors and a number of participants that use/control those floors. One of the participants is the group controller.
All communication is done by using floors. For the handling of requests, a separate mechanism exists.

The above-mentioned ideas will be used to make a generic DTC system. To simplify the design and implementation, there will be only one groupmeeting in this conference. This means that there is no need for the definition and use of conference participants or a conference controller.

3.4.2 Detailed description of the generic desktop teleconference model

The generic DT model consists of one groupmeeting that is controlled by one groupmeeting controller. This groupmeeting has a number of floors and a number of participants that can use or control those floors. These floors are each controlled by a floor controller and have a number of floor participants. Each of those floor participants may, when permitted by the floor controller, access (source and/or sink) that particular floor.

Floor controllers are group participants, assigned to control a floor by the groupmeeting controller.

A floor participant may ask a floor controller permission to access a floor. A floor controller may grant or reject such a request. When a request is granted, a floor participant instantly gains the requested access.

A floor controller can always give or withdraw floor-access rights from floor participants at will.

3.4.3 Description of the scenario, the actors and their roles within a generic Teleconference model

There is only one scenario at this moment for the generic DT model. That is the “simple teleconference” scenario. Within this scenario, 4 different roles can be distinguished. These are not the actors, since a participant can have more than one of these roles. There is just one actor, called “participant” that can play those different roles.

Actor
Participant

Roles

Groupmeeting controller:	A participant who is allowed to manage and release a groupmeeting. He can create or delete floors and assign floor controllers.
Groupmeeting participant:	A participant who has any combination of management and medium access rights.
Floor controller:	A participant who controls the access to a floor.
Floor participant:	A participant who may have or request access to a floor.

In this scenario, we have one groupmeeting controller and a number of groupmeeting participants.

3.4.4 Detailed lecture scenario for a conference system using a generic desktop Teleconference model

Groupmeeting Controller (GC):

A groupmeeting controller enters a conference and therefore (there is only one) the groupmeeting and identifies it. He can create and delete floors and assign the accompanying floor controllers. He can take part in the groupmeeting. At the end, he will leave the groupmeeting.

Floor Controller (FC):

A floor controller gets floor control rights. He thereafter controls the access rights of the floor participants to the floor and uses that to manage the floor in order for the discussion on his floor to run smoothly. At the end, he will release or lose his floor control rights.

Groupmeeting Participant (GP):

A groupmeeting participant enters a conference and identifies himself. After that, he takes part in the groupmeeting by becoming floor-controller and/or floor participant for a number of floors in order to have access to a number of floors.
He can always leave a groupmeeting.

Floor Participant (FP):

A floor participant joins a floor by asking and getting sink-access for a floor to the floor controller. He can also ask for source-access for the floor to the floor controller or release access. He can always leave a floor by releasing source- and sink-access for that floor.

A more detailed description can be found in Appendix D.

3.4.5 Rights that belong to roles

Because right are important within such a hierarchical model as this generic DT model, these are treated separately.

A Groupmeeting Controller is allowed to:
- Start and end the Groupmeeting
- Add and delete the Groupmeeting Participants to the Groupmeeting
- Add and delete Media to the Groupmeeting
- Assigns Floor Controller to media

A Floor Controller allowed is to:
- Start and end Participant's access to the Medium
- Grant or reject a Participant's request for Media Access to the Medium

A groupmeeting participant is allowed to:
- Request to Groupmeeting Controller to become floor-controller
- Leave Groupmeeting
- Leave Conference

A floor participant is allowed to:
- Depending on Medium access rights, read and write from/to this Medium
- Request the Floor Controller for Access Rights to a Medium
- Leave floor

It might be possible for a floor to have no controller at all. In that case, access to that floor might be free for all floor participants, or may be according to fixed rights.

Next, follow some diagrams that explain the relations between the different roles and components of the system in more detail.

3.4.6 Task analyses for a conference system using a generic Teleconference model

The task analysis are mostly derived from the scenarios mentioned above can be found in Appendix D.

The result of the scenario in appendix D is a global description of the cohesion and functionality of all defined roles within a generic DT model. The result of the task analysis in Appendix D is a large set of tasks that can be performed for each role within that given scenario.

3.4.7 Entity-Relationship diagrams

To describe the actors, their roles and their mutual coherence more clearly, not only point wise scenario-definitions and task analysis, but also E-R diagrams will be used.

These E-R diagrams will be presented separately for different aspects of the DTC. A stepwise refinement will be used to introduce details at a later stage. After that, a last diagram, containing an E-R diagram of the complete generic DT model in full detail, will be given.

First, Figure 6 gives a view of the generic DT model with respect to all participants, the groupmeeting and the floors

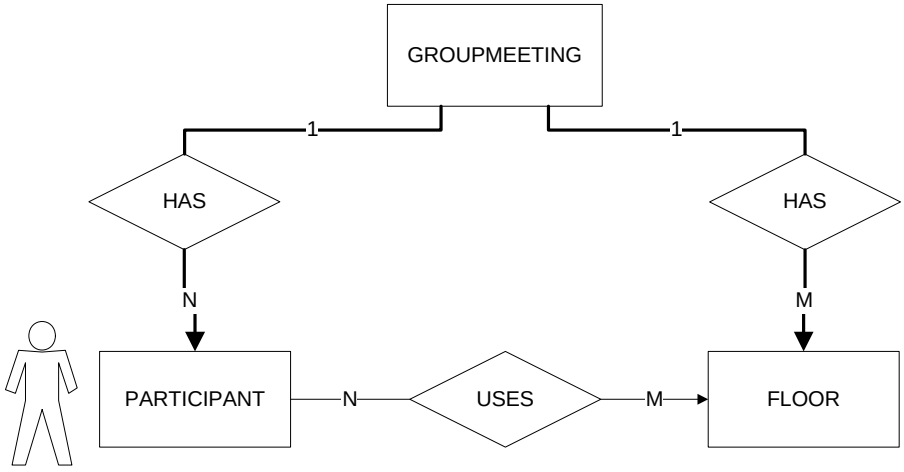


Figure 6: Simple E-R diagram of a groupmeeting.

It consists of participants who use media to communicate.

Figure 7 adds all controllers with respect to the groupmeeting and floors. A groupmeeting controller controls the whole groupmeeting and therefore may define floors and appoint floor controllers and floor participants. A floor controller may decide the access-rights to his floor for all floor participants.

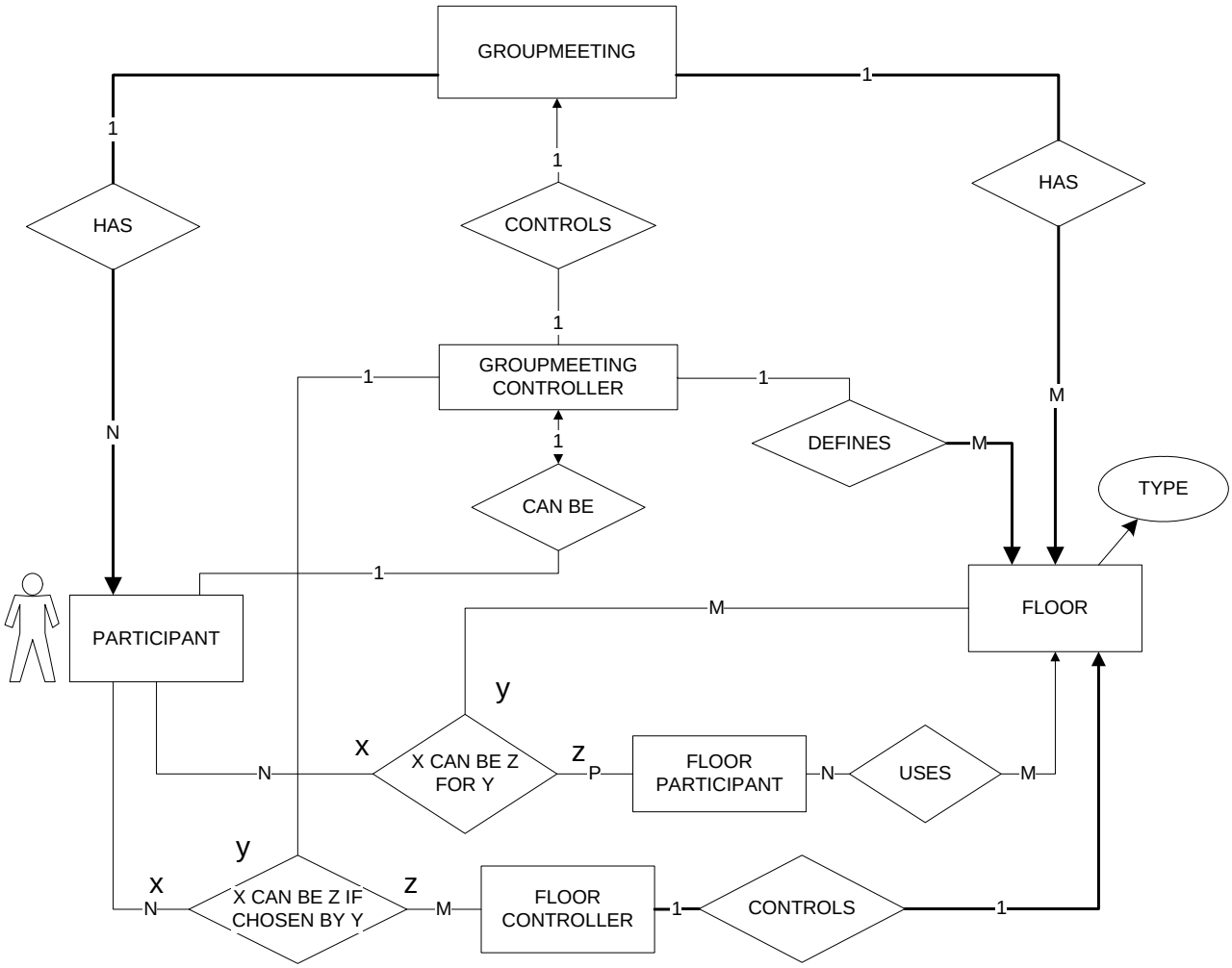


Figure 7: More detailed diagram of a groupmeeting.

A Groupmeeting has participants and floors
Each floor is of a certain type (e.g. audio, video, text)
A participant can be a groupmeeting participant. One participant can be a groupmeeting controller.
The groupmeeting controller controls the groupmeeting.
Several participants can be floor controller for a certain floor or a floor participant for a certain floor.
The floor controller controls his medium/floor.

Figure 8 gives the relation between the groupmeeting, the groupmeeting controller and groupmeeting participants in a separate E-R diagram in more detail.

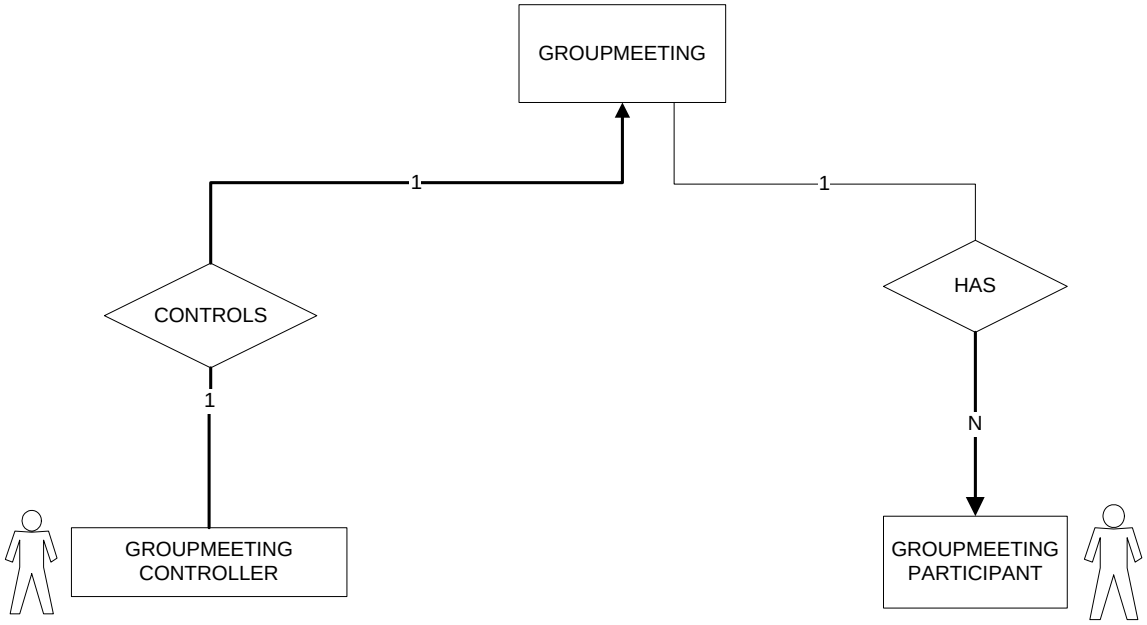


Figure 8: The relation between group controller and group participant.

There can be one groupmeeting controller and several groupmeeting participants. The groupmeeting controller controls the groupmeeting.

Figure 9 gives the relation of floors, floor controllers and floor participants in a separate E-R diagram in more detail.

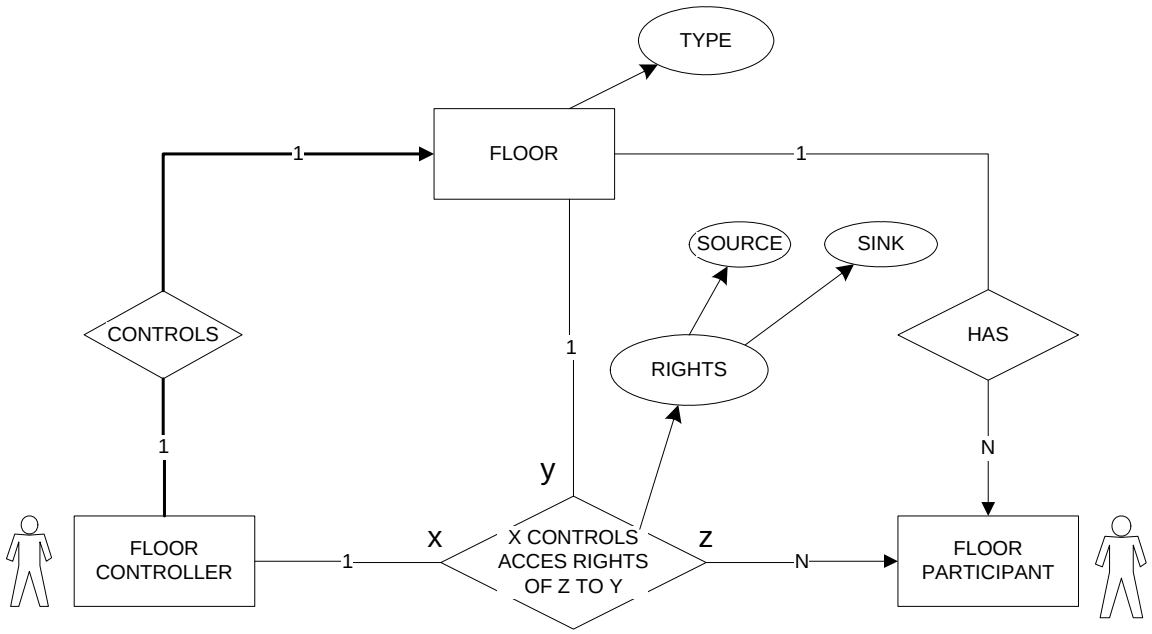


Figure 9: The relation between floor controller and floor participant.

- *A floor has a number of floor participants and a floor controller.*
- *The floor controller controls the floor.*
- *The floor controller controls the access-rights from the floor participants to the floor.*

In Figure 10, all former E-R diagrams are combined into one single E-R diagram.

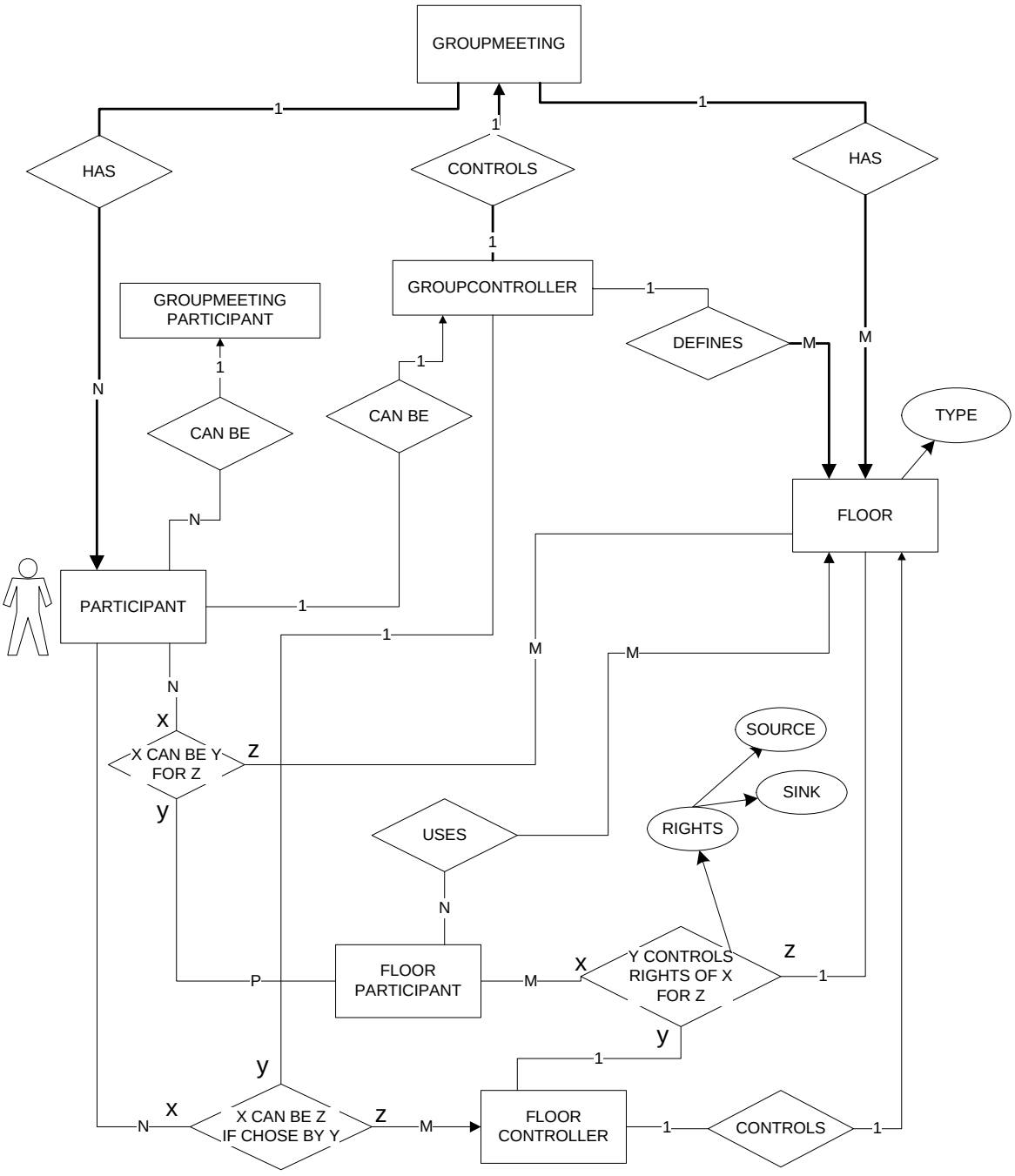


Figure 10: Total diagram.

3.5 The mapping of the DTC system to a system using a generic desktop teleconference model

In this part, we will map the roles from the generic DT model as defined in Section 3.4 onto the actors from the DTC system as described in Section 3.3. When this mapping is applied to that Generic DT model, it will give us the desired system that has all the functionality of the defined DTC system, but is more generic.

3.5.1 The mapping of the systems

In short, one can summarise the former chapters as follows:

Each participant is an actor with a single role within the DTC system. An actor within the DTC system has an educating role as a student or a teacher.

The actor, a participant, from the generic DT model has one or more roles in that DT model. To those roles belong certain rights.

These two descriptions can be combined into one description:
Each participant is an actor with a single role within the DTC system. An actor within the DTC model has an educating role as a student or a teacher and has one or more roles in the generic DT system. To those roles belong certain rights.
Figure 11 represents this in an E-R diagram.

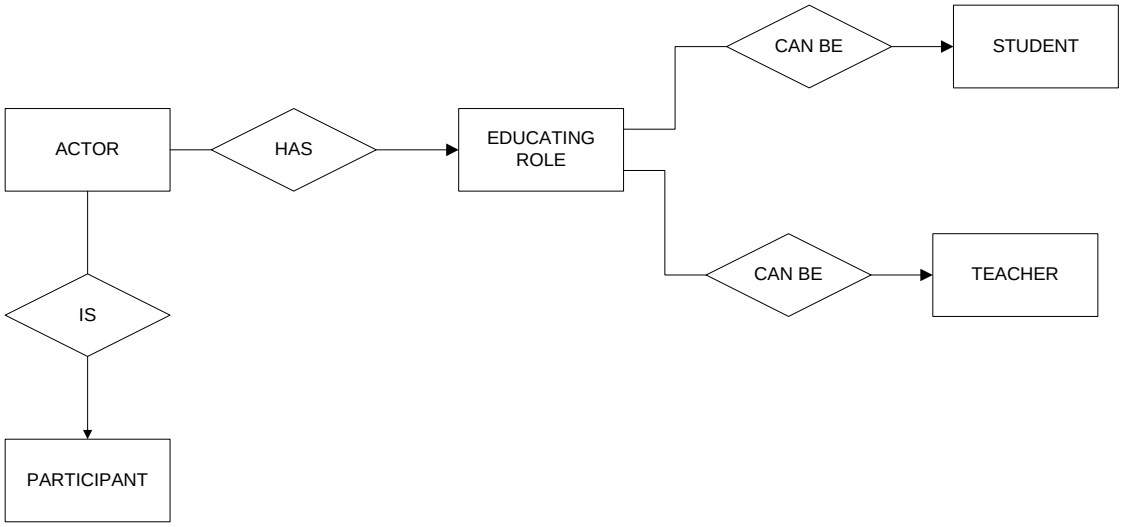


Figure 11: Mapping diagram.

A mapping diagram from the participants in the generic DT model to the actors in the DTC system

Now, this diagram can be combined with the two last diagrams of the former sections (Sections 3.3 and 3.4). This gives one large diagram in which DTC system is mapped onto the generic DT system. This is done in Figure 12.

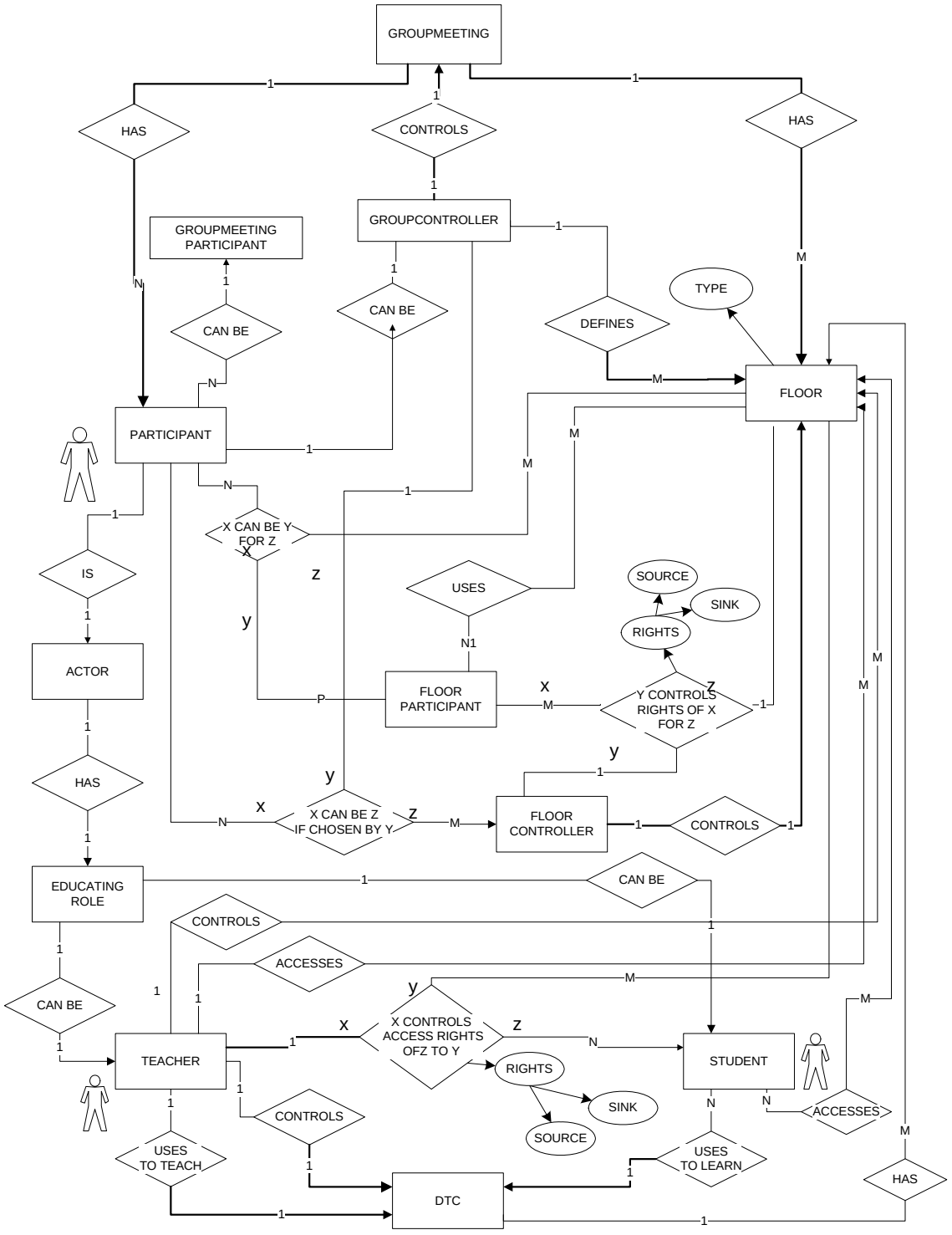


Figure 12: All combined

The diagram of the generic DT model, combined with the diagram of the mapping from the generic DT model to the DTC system, combined with the DTC system itself.

3.5.2 The mapping of the actors and roles

The mapping in Figure 12 is not yet complete. The actual mappings from actors in the Desktop Tele-classroom Conference system to the roles in the generic Desktop Teleconference system must still be determined.

If we look at the teacher/student scenario's, we must conclude that the teacher has all control over the group and over the media (floors) because he can admit or refuse students from a meeting and he is the only one that who can grant the floor to someone.

The mapping from teacher and student to roles can be summarised as follows:

A teacher is always the group controller.
A student is always a group participant.
A teacher is floor controller for all floors present.
A student is a floor participant for all floors present.

In Short:

	Groupmeeting Controller	*Floor Controller*	*Groupmeeting Participant*	*Floor Participant*
Teacher	Yes	All floors	-	-
Student	-	-	Yes	All floors

Table 2

The mapping of the roles for a desktop tele classroom to those of a desktop teleconference system.

4 HCI design for the DTC system

4.1 Introduction

The interaction of the DTC system with its users, (i.e. the whole of functionality between the users and the DTC system) has been defined as a number of tasks that those users, defined by their roles, can perform within the DTC-system (see Chapter 3).

The HCI for the DTC system must provide that interaction with its users by offering a number of functions to perform all those tasks. These tasks, as in Chapter 3 defined, will therefore guide the design of the HCI.

These tasks are dependent on the role that a particular user has. Therefore, also the functionality that the HCI offers can differ for each user, depending on the role that the user has within the system. For instance: a normal student must not be allowed to prevent everybody from speaking.

The actual design is meant for a DTC system, but a number of design-features for a more generic desktop conference system [as described in Chapter 3] are kept in mind because the DTC system can be extended.

4.2 General overview of the functionality of the HCI from the DTC system

4.2.1 Generic conference system functionality

When we look at the scenarios and task analyses from Chapter 3.4 and Chapter 3.5, a number of generic functions can be distinguished.

- When a user starts or enters the system, he first has to identify himself and the conference he wish to join. In this procedure, the user not only assigns himself a unique identity that can be used throughout the system. He also defines his initial role and the accompanying initial rights for that session within the DTC system.

- The HCI must provide control functions for each local medium like audio or video. In addition, it must provide functions to use each floor that has been defined.

- Another general function that the HCI has to offer is disconnection from the groupmeeting and return to the enter-groupmeeting part.

- When a user has a role as a groupmeeting controller, the HCI has to offer functions to control the groupmeeting. This means that the HCI has to offer functions to create floors and to assign control of those floors to groupmeeting participants that thus become floor controllers.

- When a user has a role as groupmeeting participant, the HCI has to offer functions to request and return control of a floor.

- When a user has a role as a floor controller, the HCI has to offer functionality to control the assigned floors. This means that the HCI has to offer functions to enable read and/or write access for one or more of these floors, if requested.

- When a user has a role as floor participant, the HCI has to offer functions to request and return read and/or write access for one or more floors.

- When a user has a role as a controller (floor or groupmeeting), the HCI has to offer a list of all participants present, to give the controller a general overview.

The HCI that is being designed here is not meant to be a complete generic conference system, but must at least offer the functions of a DTC system as described in Chapter 0. This means that not all functionality as mentioned above has to be implemented in this cycle.

4.2.2 DTC system functionality

Mapping

The functionality of the HCI for the DTC system is smaller than that of the generic DT model because there are only two actors defined: Teacher and Student.

Still, some provisions must be made to keep the HCI more universal. This has the advantage that the functionality of the DTC is easier to extend during a later design cycle. Therefore, a restricted implementation of a system according to the generic DT model as described in Section 4.2.1 will be designed with a mapping of generic roles to specific DTC-actors.

In a Generic DT model, as described in Section 3.5.2, the teacher has the roles of groupmeeting controller and of floor controller for all floors. A student has the role of groupmeeting participant and of floor participant for all floors. Groupmeeting will be the same as “classroom”.

Restrictions

Since the teacher is both groupmeeting controller and floor controller for all floors, it is not necessary for the HCI to offer functions to assign floor control to groupmeeting participants. The teacher is only concerned with floor control. Therefore, the decision was made to implement group control functions only as a mock-up.

Because each student has to be able to follow a lecture using all available floors, it is not necessary to provide functions to request read access for floors. Instead, sink-, or read access-rights should be given to a student by default for all floors. The decision was therefore made to let all users always have sink-rights.

Since all students always have sink access is enabled by default for all floors and since the Session layer implementation does not provide function for deletion of users from floors [2], there are no functions to delete users from all floors and therefore no functions to delete users from the groupmeeting.

4.3 The mapping from tasks to functionality

In order to design the functionality of the HCI for the DTC system according to a limited generic DT model, we first have to map each task from the task analyses in appendix D to functions that the HCI has to offer to the user. This mapping is done in appendix E and results in a list of functions that, when implemented, the HCI must offer to the user.
Those tasks and functions are related to roles in a generic DT model. Later, those DT-roles can be mapped onto roles from the DTC system according to the mapping as given in Section 3.5.2.

4.4 The different HCI main-elements of the DTC application

4.4.1 Division into HCI-views

To give the user a better understanding of the functionality of the system and to enable the user to find a desired function more quickly, the HCI will be split up in different parts. These each represent a different logical "block" of functions that have a strong cohesion. These parts are called "HCI-views". Each of those HCI-views will get a separate, coherent, visual representation within the system because of the coherence of its functionality. A user can, according to the defined relations between those HCI-views, switch between them.
This approach is illustrated within the Platinum documents [4].

4.4.2 The HCI-views

From the former section (4.2), we can derive de following main HCI views.

1. A HCI view for entering a groupmeeting.
2. A HCI view for inspecting the groupmeeting.
3. A HCI view for modifying the groupmeeting.
4. A HCI view for inspecting access to a number of floors
5. A HCI view for modifying access to a number of floors
6. A HCI view for requesting access to a number of floors.
7. A HCI view to control local media.
8. HCI views for adapting and using floors.

Hereby the decision was made to combine the functionality of inspecting, modifying and requesting. This is done for both groupmeeting and floors. This is the case from the second HCI view up to the sixth HCI view. This is done because, in the generic conference system, a floor controller can also be a floor participant for another floor. This means that, when not combined, there would be two separate HCI-views that provide functions on the same subject (i.e. floor-access or floor-control). This would be confusing for the user and use a larger part of the desktop. Furthermore, the combination of the two is useful for a floor participant because he can have knowledge of requests, even when he is not the one in control. This can influence his decision to make a request.

This results in the following main HCI views

1. A HCI view for entering the groupmeeting
2. A HCI view for inspecting and modifying the groupmeeting
3. A HCI view for inspecting, modifying and requesting access to a number of floors
4. A HCI view to control local media.
5. HCI views for adapting and using floors.

4.4.3 Relation between the HCI views

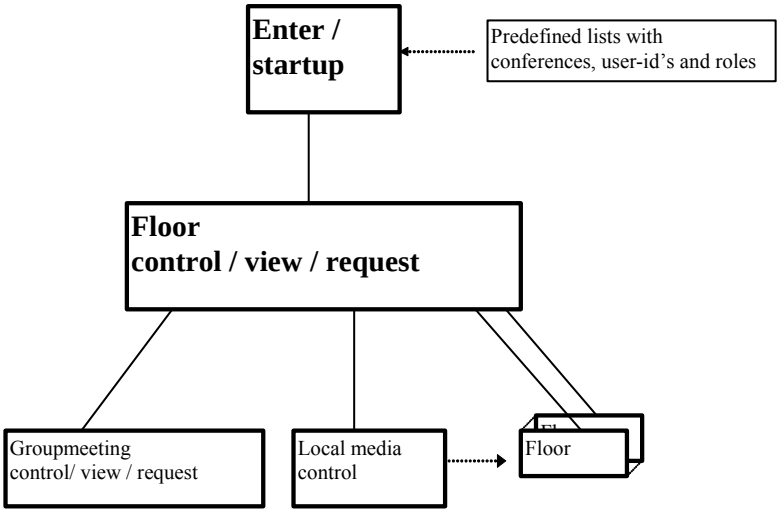


Figure 13: How the main parts of the HCI are bound together.

The user always starts at 1 (enter groupmeeting), where the user, his role and the groupmeeting will be defined.

As central HCI view, the floorcontrol HCI-view is the best choice. This is mainly the case because it gives an overview of the groupmeeting and provides the very basic tools for a DTC system for both the teacher and the student. Such a central view will be presented immediately after the start-up HCI-view. The HCI-view for groupmeeting control is higher from a hierarchical point of view and therefore seems also a logical choice. However, this will not be used very often, even in complete system according to a generic DT model, because control of floors is not likely to change quickly. In this case, for the DTC system, the second HCI-view will hardly be implemented or used. The HCI for local media control will probably be used often, but is very low from a hierarchical point of view and therefore is not the best choice to use as a central HCI-view.

From within the floorcontrol HCI-view, because it's central positioning within the system, the other HCI-views can be called. In addition, from this HCI, the user can return to the first HCI-view (enter groupmeeting).

These other HCI-views deal with groupmeeting control, local media control, and floor-access. Local media control will take care of properties for audio and video and therefore has a direct effect on some floors. The HCI-views for floor-use deal with the actual use of floors, that have media like text-chat or a whiteboard that need their own HCI, or the access-control to media on a per-floor base.

This is all illustrated in Figure 13.

4.5 The fully designed HCI-views within the DTC system

4.5.1 Considerations for all HCI-view descriptions

All demanded functionality as defined by the function as derived from task analyses in Appendix E, will be divided over the different HCI-views.

- Per function, a short description of how the HCI will handle this will be given.
- A number of functions will not be fully designed yet. These are given in a separate section.

A number of very general functions will always be available:

Status field
In order for error messages to appear, a kind of status field with continuous changing messages is necessary. This status field would appear in all larger windows so the user always knows it when problem has arisen.

Quit/hide function
To return to a former level of the HCI or to clean up the desktop, a user must always have the possibility to quit or hide a particular HCI-view.

4.5.2 A HCI view for entering a groupmeeting

This must be one single basic entry-window that acts as a kind of gateway to the rest of the system. The following functions must be offered according to the task analyses in appendix C and D.

Entering a unique personal identification number
This will be done in a normal input field.

Choosing of a groupmeeting
Here, the best solution would be an input field, combined with a pull down list to chose or type the name or number of the desired groupmeeting.

Join conference
This is just a button.

user-ID check /college ID/ roles
The user identification number and groupmeeting identification must be entered or chosen and must match each other before a conference can be joined. When a conference is joined, the match of identification number and conference determines the user's role. When a conference can't be joined, an error message must appear.

In Figure 14 is the visual appearance given for such an HCI view.

Figure 14 The login HCI-view.

4.5.3 A HCI view for inspecting / modifying / requesting floors

Assuming that all controlled floors are represented in one HCI view, the following functions must be offered according to the task analysis for the different roles.

General presented information

Since the functionality that the HCI offers to the user can differ for different roles, there should be an indication for the role that the particular user has. In addition, an indication of the ID and name from both the user and the joined groupmeeting should be present.

Presentation of all floor participants

This view is presented as a central view from which other views will be called. This means that this HCI-view must have the functionality to call all other HCI-views. This can be done with the help of buttons and menus. It also means that this HCI-view must present the before mentioned general information.

The presentation can be done in a number of ways. The most common ones are a table and a list. Within a table, the requests and rights can be shown or specified for each participant and for each floor separately. This gives the most complete view of the groupmeeting, especially when one is controller for more than one floor. The disadvantage of such an approach is that it is a complex overview, especially when the numbers of participants and floors become larger. The use of a simpler, scrollable, list of participant's names or ID's with some extra information is easier to use. For this specific application, a DTC system, this is even more the case because there is only one controller, holding all floors, who only gives floor-write rights occasionally. Therefore, we have decided to use a scrollable list of groupmeeting participants.

The ID that was used to enter the groupmeeting is a unique number for each person. For the presentation, these numbers must be replaced with the names of each participant because in practice, names are used to address people. It must be possible to display both name and number because two people might have the same name, but never the same number.

Menu-bar

To comply with visual HCI-standards (see [7]) and to offer the user an option to use this central HCI with a keyboard, a menu-bar is needed. This will offer, when possible, the same operational functionality as the visual HCI described below. It should be organised and grouped in the same ways as the rest of the HCI-window.

An example of an unfolded menu is given below.

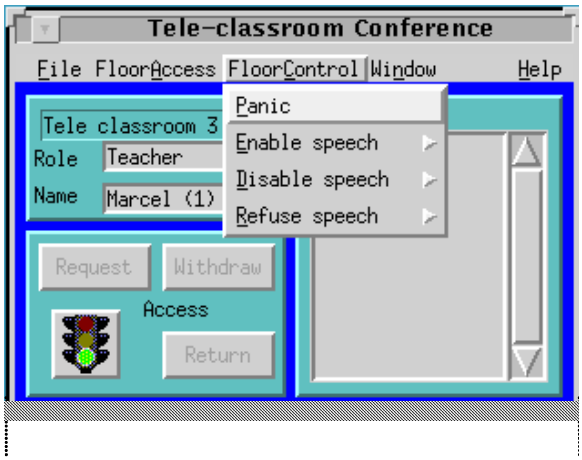


Figure 15 An unfolded menu.

When the actor is a floor controller:

Showing request from floor participants for write access to certain floors

Incoming requests contain not only the ID of the sender, but also the ID's of the floors that are requested and a request message. This message can contain, for example, the subject of the request.

To present these requests, there are a number of possibilities:
- Popup windows with information regarding requesting participant that are presented directly at the time the request arrives.
- An extra list with all requests in requesting order with extended request information like requested floors and request message.
- The marking of requesting participants in the existing participants list, by using colours and / or special characters (with or without presenting extended request information).
- A Popup window with extended request information like the request subject, that is called at will.

The disadvantage of the first alternative is the intrusive and interrupting manner of presenting requests. Using only the second alternative gives no real direct survey of the whole "lecture room". When the number of participants becomes larger, the normal list can't be totally seen without scrolling, so requests can be missed by the controller. Using only the third alternative gives no information about the order in which requests were filed. The fourth alternative gives no means of direct request-indication and must therefore always be combined with one of the other three methods. Even then, it gives no means of directly surveying extended request information like the request message.

The decision was made to use a combination of the second and third alternative. The second alternative presents a direct and summarised survey of the whole "lecture room" or groupmeeting, while the third alternative gives a direct, in-depth survey of all pending requests that retains the order the requests were filed. Actually, within the function-analyses (see appendix E) a participant-list and a request-list are already mentioned, but more for ease of expressing certain functionality.

Following the above, it was also decided that the presented request-information should be short for the participants-list to keep that list physically narrow and at most contain the requested floors. Within the request list, all request information, the requested floors and the request-subject, should be present.

Granting requests for floor write access for all requested floors

Since the choice was already made to use lists for both all participants and participant-requests, it was decided that this would be done based on a selection in one of those lists as defined above in combination with a popup menu. In this design cycle there is only a possibility for a fast acknowledge of request, granting all the requested media.

Giving (granting without a previous request) floor write access for all floors

The easiest and therefore chosen option is to integrate this with the above function of granting of floor access requests. This can be done by presenting the same popup menu as described above in the participants-list when the selected participant has not issued a request. In this design cycle only a possibility for fast acknowledge of requests should, in this case, just grant all existing controlled floors at once.

Granting floors for now means only granting source rights for a floor, since sink rights are always enabled by default.

Denying requests for floor access

This can best be done in the same way as granting requests, based on the lists already present with help of a popup menu, coupled to the present list selection. This has the advantage that it presents a uniform method to the user of manipulating all floor rights of a participant.

There should be a possibility for a fast refusal of a request, and a slower, but more extended possibility using a popup window in which a kind of message / reason for the refusal can be given to the requesting floor participant.

The fast refusal option is particularly useful to "clean up" the request list after a question about a specific topic has been answered and more questions about the same topic still remain but have already been addressed.

Grabbing floor access

There are two different types of floor grabbing possible:
- Grab one or more floors from a single participant.
- Grab one or more floors from all participants.

For the first type, we have decided to use the same method as for granting requests, using the participants and request lists to select participants to grab the floor from. This has the advantage, as already mentioned above, that it presents a more uniform method of manipulating the floor rights of a participant.

There should be a fast possibility to grab all granted floors from one participant and a possibility to grab granted floors from all participants at once. This can be done using a button or menu-option and a more extended possibility, using a kind of popup window, to determine precisely which floors should be grabbed.

For the second type of grabbing, from floors from all participants, there is no need to select a participant, only the floors to grab can be selected. It is possible, like for the first type, to give two possibilities: A fast one, grabbing all floors at once, and a more precise but slower one, using a popup window in which the user can determine the floors to be grabbed.

The fast possibility might be to readily invoked by accidentally pushing the wrong button. It is better to have just the slower option, because a user has the possibility to cancel the operation. Therefore, we have decided to allow only the slower option.

Show floor access for all floor participants

This can be done in both the participants list and the request list, as described above, by using colours or special characters. In addition, it can be useful to give an indication of the floors involved.

When the actor is floor participant:

Request to gain write access for certain floors

Here are a few possible options:
- A fast request button, allowing all floors to be requested at once.
- A slower popup window that allows the selection of the floors to be requested and offers the possibility to give a request-message containing for example, the subject the request is about.

For simplicity, both for the HCI definition and for a user so he has not to keep track of a multitude of requests, it is decided to allow only one pending request at a time.

Within the DTC system, within this design-cycle, there can only be one floor controller that controls all floors at once. To keep this simplicity of one pending request, it is decided to allow only requests for combinations of floors that are controlled by a single floor controller.

A fast request button, requesting all floors at once, is not always possible since there could be, in a next design cycle, more than one floor controller. Besides, making a request should in many cases be made with consideration and with the expression of a request-subject. Therefore it is decided to use only a more extended, but slower request popup window. Here the floor participant can define the floors and their properties to request and a message or subject to be displayed to the concerning floor controller. The limitation for the possible combinations of floors can be forced by only presenting allowed combinations of floors in a floor-selection list.

Show status of request

A pending request will be represented by a kind of "traffic light" that marks the state of the requested floors. The reason for this simple indication is that it gives the user a simple and quick way to view his global situation. This is especially useful in the situation of our DTC system because Floor Participants will, in most cases, request all floors at once, or just one single floor, to pose their question and return it immediately afterwards.
This description can be further simplified and depends on the fact that all sink-rights for all floors are enabled by default. These rights, therefore, have not to be shown within this global status. This can be done in the following way with a traffic light:

Red: The participant has no floor source-rights at all and no pending request. When the light jumps to this colour that can therefore mean that a grab or a return occurred for all granted floors or a request was refused.
Yellow: The participant has a pending floor source-access request.
Green: The participant has at least source-access for one controlled floor. When the light jumps to this colour, it means that a floor-request was granted or a spontaneous grant occurred for access to at least one floor. An example of a traffic light in the "green" position is given below.

Figure 16 Example of a green traffic-light.

A more complete status will be given in a separate popup-window. This window must contain a list of all floors and their corresponding control-, sink- and source- rights.
To keep a similarity with the rest if the HCI, it is decided that this is also to be done by using traffic-lights that have the same kind of meaning as mentioned above: red means no right, yellow means requesting, and green means that rights are present. Below, the visual appearance of such a window is shown.

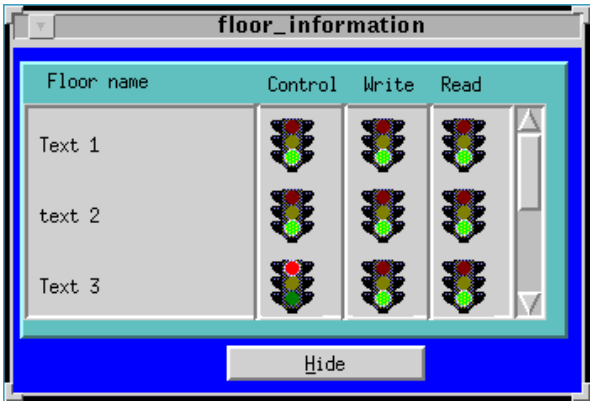


Figure 17 Floor-status HCI-view.

Withdraw a request

This can best be done by using a withdraw button, to withdraw the pending request.

Return floor access to floor controller

This can best be done by using a return button, to return all granted floors.

Show floor access indications

This is partly integrated with the “traffic light”. This light shows whether or not write access for some floor has been given.

There should be a popup window that gives information about the access-status for each floor separately in a list.

For each different floor, there should be a message within the HCI-view for that particular floor, indicating access permissions. In case of audio and/or video, it has been decided that this will, for now, be done within the HCI for local media control. This has been done because there can only be one audio and one video-floor and therefore separate HCI-views for those floors are unnecessary or not possible yet.

In Figure 18 is a visual appearance given for this HCI view.

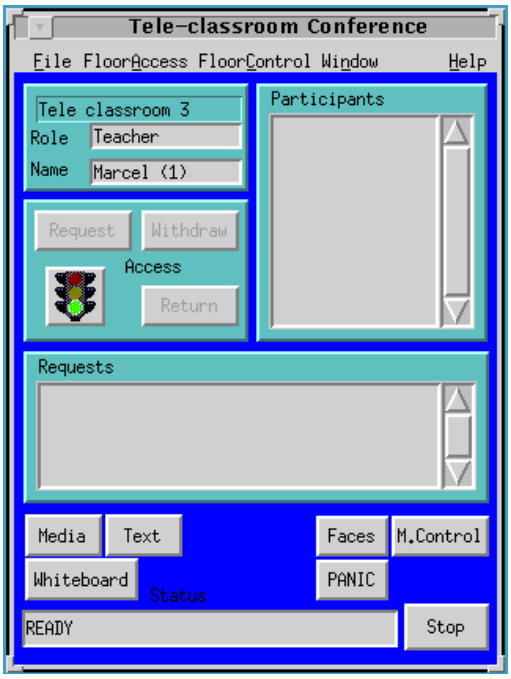


Figure 18 The floorcontrol HCI-view.

4.5.4 A HCI view for local media control

Local media controls all local properties of standard multimedia devices like audio and video. All these general properties apply for all floors that use audio or video.

For all types and directions of local media, there should be an indication whether they can be used or not. This is dependant on floor access-rights. This gives a participant a simple indication what media might be used directly and what media might need to be requested.

To prevent a multitude of windows, that each have to be managed (placed and sized), a good choice will be to integrate all audio- and video-controls within one window that exists out of multiple, distinguishable parts, dependent of device and direction. To keep the size of this window within limits, an option should be present to shrink parts that are not directly needed.

The multiple parts are:
- Video In
- Video Out
- Audio In
- Audio Out

Furthermore, there can be a number of functions, bound to each video-window or participant

Video in: Properties of video in

Enable / disable all video reception

It should be possible to disable video reception because the necessary video windows take up space on the desktop, use processor time and the reception of video increases the network-load enormously. To keep this option simple, this will be a general function to enable/disable all video reception at once. This can be operated by a using simple check button, that can be pressed in an “on” and an “off” status.

Status video reception

This is an indication, whether or not it is possible and allowed to receive video.

Video out: Properties of video out

Enable / disable video transmission

For privacy reasons it should be possible for a user to block transmission of his/her video-image to others. This function will enable or disable all allowed video transmissions. This can be operated by using a simple check button.

Enable / disable video transmission preview

It should be possible to enable/disable the preview window for outgoing video. A preview window takes up space on the desktop and uses processor time, but it can be very useful to check your physical position within the camera-frame. This function simply enables or disables the video transmission-preview window. This function can be operated by using a check button.

Size

This will influence the transmission resolution (and therefore size) of the video stream. This property can have multiple, ordered values, so best solution will be to use a slider bar to adjust this value. It must be possible to adjust this setting because it concerns the real resolution of the transmitted image. This will influence the network load and computing load of the transmitting machine and all receiving machines. It determines the maximal resolution that can be used for viewing the incoming video stream on the receiving side.

Brightness

This concerns the brightness of the transmitted image. This can be done by using a slider-bar within each video window. This has the advantage of adjusting the view more according to available lighting within the surrounding.

Status

This is an indication whether or not it is possible and allowed to transmit video.

Audio in: properties of audio in

Enable / Disable

This function is needed since it must be possible for a user to mute incoming audio e.g. to be able to speak to others in the room. No audio will be received so it is best to make it a function to enable/disable general audio reception to diminish the network load. This can be a simple check button.

Volume

This Function adjusts the general play volume of the audio device for all incoming audio from all floors and all participants. This can be done by using simple slider bar.

Destination

This function will consist out of a number of radio-selection buttons. Only one can be pressed at a time. This is needed because there should be a possibility to change the audio destination e.g. to use headphones instead of the internal speakers.

Show audio-in status

This is an indication whether or not it is possible and allowed to transmit audio.

Audio out: properties audio out

Enable / Disable

For privacy reasons it is desirable for the user to be able to block transmission of audio to others. This function enables or disables all audio transmission. This function can be performed by simply using a check button.

Volume

This Function adjusts the general play volume of the audio device for all incoming audio from all floors and all participants. This can be done by using simple slider bar.

Source

This function will consist out of a number of radio-selection buttons. Only one can be pressed at a time. This is needed because it can be necessary to change the audio record source to allow transmission of other audio source then a microphone (e.g. from a tape or a CD-player).

Show audio-out status

This is an indication whether or not it is possible and allowed to receive audio.

In Figure 19 is the visual appearance given for such an HCI view.

Figure 19 The media HCI-view.

4.5.5 HCI views for adapting and using floors

Audio

This will be done directly by the network layer for performance reasons.

Video

Display video in and display preview out

This will be done directly in a window by the network layer for performance reasons. At the moment, these are separate windows without any extra control.

Text

Show incoming text

For text channels, it is best to have a separate HCI component for each channel, since they all have their own history and can be easily separated. The advantage of separate windows is mostly, that they can be minimised or hidden individually, when they are not needed.

Like in many existing chat applications, this can best be done within a scrollable text window that keeps a history of all incoming text. Since the text can originate from more that one user, it has been decided to put the name of the sender in front of each received line of text. There should be a possibility to clear this history.

Input and send text

This can be done by using a simple input-line that is cleared each time the text is transmitted.

It has been decided to use a combination of an input-line together with a history-window. This is better than the other considered option, an input text window combined with an output-text window. This is because, for the user to keep an easy track of a conversation with multiple participants, it is best to include all transmitted lines as well within one window, the reception text window.

In Figure 20 is the visual appearance given for such an HCI view.

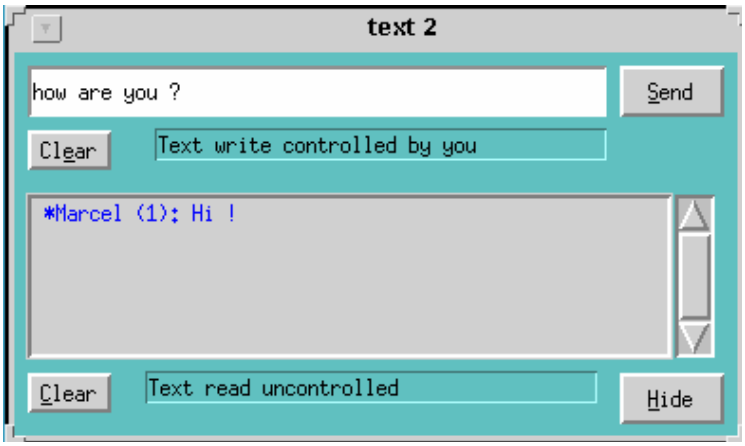


Figure 20 Text-window.

Display video in and preview out

This will be done directly in a window by the network layer for performance reasons.
At the moment, this are separate windows without any extra control. All controlling has to be done external by a separate part of the HCI.
In this design-cycle, due to functionality of the used network layer, there can only be one video channel and therefore only one video-floor.

Audio in and out

This will be done directly by the network layer for performance reasons.
In this design-cycle, due to the functionality of the network layer, there can only be one audio channel and therefore only one audio-floor.

4.6 Suggestions for a next design-cycle (half-implementations, mock-ups and proposals)

The first complete design cycle for a DTC system, was done by Vincent Korenromp [1]. This HCI for a DTC system is part of the second design cycle, together with the session layer implementation as designed by Alex Jongman [2]. In this present design and implementation cycle of the system concentration will be on audio and text due to the limited functionality of the implemented session-layer. Video support will be implemented, but might not be complete. In addition, more than one floor of each type of medium for audio and video is not possible yet.

In the future, this system will probably undergo a third design cycle, adding multiple platform support, adding more floor types and giving more audio and video connection possibilities.
In addition, there will be extra additions to extend the functionality of this DTC. This must be taken into account while designing and implementing the HCI for the Desktop Tele-classroom Conference system. An example of such an extension is the coupling of this DTC system to a larger controlling educational system and the use of central databases with educational information [5].

Next there follow a number of suggestions for HCI-designs in such a third design cycle.

4.6.1 A HCI view for inspecting, modifying and requesting floors

When a user is floor participant

Showing request from other floor participants to have write access to certain floors

It can be very useful for all participants to see all pending requests. This can aid their decision whether or not to initiate a request themselves.

When a user is floor controller

Granting requests for floor write access for one or more floors

Granting all requested floors at once can be done with a popup menu, but in the future, there could be a possibility for a slower, but more extended possibility using an extra popup window in which floors can be assigned at will. The popup window will open with all requested floors enabled, but allows the floor controller to modify this.

Giving (granting without a previous request) floor write access for one or more floors

Granting all requested floors at once can be done with a with a popup menu, but in the future, there could be a possibility for a more slow, but more extended possibility using an extra popup window in which floors can be assigned at will. The popup window will open with all requested floors enabled, but allows the floor controller to modify this.

Sink-rights for floors

Granting floors meant only granting source rights for a floor, since sink rights were enabled by default. In the next design cycle, this could change so that not only write access can be granted, but read access as well for floor participants.

More Floor Controllers for different floors

It might be possible that there is more than one Floor Controller. Each Floor Controller then controls a different set of floors.

This would have some implications: a controller would be able to be a Floor Controller and a Floor Participant at the same time, controlling floors and requesting floors at the same time.

This means that the “traffic-lights” must be adapted, so a Floor Controller knows that he is a controller and therefore always has source-rights for some floors, but also can request and return floors. The choice is made to retain only a single light. To indicate control, the light should be green, but the light should also be red or yellow at the same time to indicate no access-rights or a request. When access is granted for the requested floors, the light becomes green for all three lights to indicate both access to owned floors and requested floors is possible. Examples are given in Figure 21.

Figure 21 Traffic lights with more than one light on.
Traffic lights indicating no permission, pending request en granted request for some floors, while others are controlled by the same user.

4.6.2 A HCI view for inspecting and modifying the groupmeeting

This HCI view is not actually necessary to implement a DTC system. Since the Teacher is floor controller for all floors and groupmeeting controller, there is no need to modify the groupmeeting except to delete students from the groupmeeting. This view is now given because, in the future, there might be other roles than student and teacher.

Similarity with the floor HCI-view

For floor-access control, it is easier to use a list with floors. This list would have, for each floor, the name of the floor controller behind it, since each floor has exactly one controller (or none).

The main disadvantage of using this list type is that users might get confused by totally different HCI designs. Therefore it is best to keep this HCI view as close as possible to the HCI view for inspecting, modifying and accessing floors as described in Section 4.5.3 and use a list that contains all groupmeeting participants together with a floor control request list. Actions on users, like making someone floor controller, can be performed using popup menus from within those lists.

General presented information

Since the functionality that this HCI-view offers to the user can differ for different roles, there should be an indication for the role that the particular user has.

In addition, an indication of the ID and/or name from both the user and the joined groupmeeting should be present in this HCI view because this view is used to control this groupmeeting.

Floor control indications

Within the floor-HCI view (Section 4.5.3) an indication of which participants have granted rights is done in both the participants list and the request list described above using colours, special characters and an abbreviated indication of the floors involved. This can be applied here also.

When the actor is a teacher
(Therefore, the actor is also the groupmeeting controller)

Presentation of a list of groupmeeting participants

This can be done by a simple, scrollable, list with participants name and ID together with floor control-information. This is the best way to keep a good similarity with the floor-HCI.

Deleting of groupmeeting participants from groupmeeting (denying access)

The already present participants list can be used to do this. A participant can be selected within the list, after which a pull down menu appears to perform the action 'delete'. This has a good similarity with the floor-HCI view as described in Section 4.5.3.

Showing request from groupmeeting participants to become floor controller for certain floors

Within the floor-HCI view (Section 4.5.3), this can be done by a combination of an indication in the existing list and by using an extra request-list with more in-depth information. This is also possible here. The same reasons as mentioned for the design of the floor-HCI view for having an extra request list apply here. In addition, the similarity between HCI views is important.

Granting requests for floor control rights

Within the floor-HCI (Section 4.5.3), granting or denying of requests is done by a pull-down menu from the participants list or the request list, so it is best to implement it here likewise. There should be a possibility for a fast acknowledge of request, granting control for all or the requested media, and a more slow, but more extended possibility using a popup window in which floorcontrol can be assigned at will.

There is one difference however with the floor-HCI. There can only be one floor controller for each floor, so an assignment of floor-control rights to a participant can conflict with already granted floor-control rights to another participant. In that case, the best option is to give a warning and abort the operation. This is better than just go ahead and first grab the control-rights from the former floor controller because it has large consequences for the former controller and can better be explicitly done by the groupmeeting controller.

Giving (granting without a previous request) floor control rights for one or more floors
This can be done similar as in the floor HCI view (Section 4.5.3) by integrating it with granting/denying requests. The only difference is that the use of "fast acknowledge" is useless, because it will probably almost never happen that someone gets all floors to control at once without first giving a request for this.

Denying requests for floor control

This can best be done similar as in the floor HCI view (Section 4.5.3). This is in the same way as granting requests, based on the lists already present with help of a popup menu, coupled to the present list selection.

Free a floor of all control

This can best be done by adding a virtual participant to the groupmeeting participants list like “NO CONTROL” that gets all floors that are controlled by no one. This is easy to integrate within the existing HCI definitions while there is no need for separate functions with extra buttons / menus.

Grabbing floor management for one or more floors

This can best be done based on one of the lists already present with help of some kind of popup menu, coupled to the present list selection. There should be a possibility for a fast grabbing of all floor control rights from one participant, and a more slow, but more extended possibility using a popup window in which the floors to grab can be selected.

Groupmeeting control is not as dynamic as floor control. Therefore it is not necessary to implement an aggressive function to grab all floor control at once, like there is for a floor controller to grab all controlled floors at once.

When the actor is a student
(Therefore, the actor is a groupmeeting participant)

Presentation of a list of groupmeeting participants

This will be the same as for a floor controller.

It might be useful to have a view of the request lists for all groupmeeting participants as well, so the nature of all pending requests is also known to them.

Request to become a floor controller for certain floors to group controller

It is, like within the floor-HCI view (Section 4.5.3), the best option to use a request popup window. In there the groupmeeting participant can define the floors to request control for and a message / subject to display to the groupmeeting controller.

.

Withdraw request

This can best be done by using a withdraw button, like within the floor-HCI view (Section 4.5.3).

Return floor control to group controller

This can best be done, like within the floor-HCI view (Section 4.5.3), by using a return button, to return the granted floor control rights.

Show floor control status

A pending request can be represented by a kind of "traffic light" that marks the state of the requested floor control rights in a similar way as within the floor-HCI view (Section 4.5.3):

Red:	No floor control rights at all and no pending request. When the light jumps to this colour that can therefore mean that a grab/return occurred of all granted floor control rights or a request was refused.
Yellow:	Pending floor control request.
Green:	At least control rights for one controlled floor are present. When the light jumps to this colour it means that a floor-request was granted or a spontaneous grant occurred for floor control for at least one floor.

In Figure 22 is the visual appearance given for an example of such an HCI view.

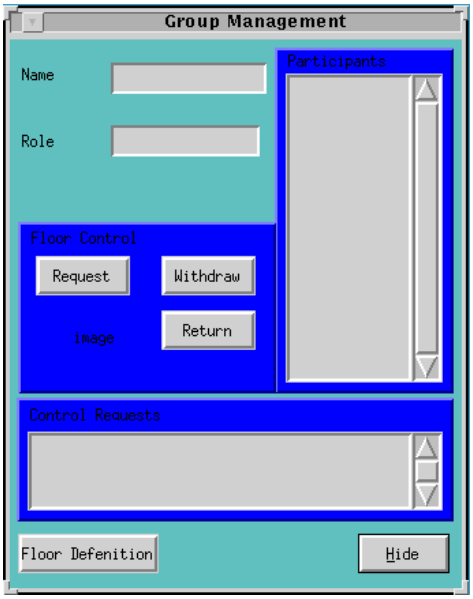


Figure 22 Group management HCI-view mock-up.

4.6.3 A HCI view for local media control

Video out: Properties of video out

Source

This function will consist out of a number of radio-selection buttons. Only one can be pressed at a time.
There should be a possibility to change the video record source to allow transmission from e.g. videotape instead of a video camera. In addition, there should be a possibility to transmit video, already present as digital recording, since this will become increasingly possible (e.g.) a video CD.

4.6.4 HCI views for adapting and using floors

Audio

Different floors

To adjust properties for a certain audio floor, it is best to make a separate popup window that can be called from the central HCI view in which the floor can be chosen to change the settings from. Since the network-layer can only support one audio-floor, this part will only be briefly treated and not implemented.

Enable / disable for a certain audio floor

This will be a simple check button

Adjust volume for a certain audio floor

This will be a slide bar.

Video

Different video floors

To adjust properties for a certain video floor, it is best to make a separate popup window that can be called from the central HCI view in which the floor can be chosen to change the settings from. Since the network-layer can only support one video-floor, this part will only be briefly treated and not implemented.

Enable or disable transmission / reception for a certain video floor

This will be a simple check button

size

This should be integrated as a pair of larger-smaller buttons within each video window (both for transmission preview and for reception). Not as slider bar because that would change size when the window does. That would be confusing for the user.

This is not the real resolution of the received or transmitted image. That will be shrunken or enlarged to fit the window-size.

Brightness

This should be integrated as a slider-bar within each video window.

Enable or disable

High detailed reception control (disable / enable per participant) is useful to prevent the whole desktop from being cluttered by video windows and to raise performance by preventing to may live video-windows at once. This could be bound to each video reception window by using a simple enable/disable check button. After disabling, the image freezes and can be minimised.

Whiteboard

A whiteboard is a kind of draw- and presentation- window and can be used to make drawings or to present sheets on which can be drawn. Others, when in possession of write rights, can add their own drawings, or even present their own sheets.

For each whiteboard channel, there should be a separate HCI component.

It should have following functions:
- Specify a file, or a number of files, to get sheets from. These could be stored in a number of formats like GIF, Jpeg, or Postscript. These sheets are presented as a kind of background, on which can be drawn.
- A “next”, a “previous” and a “refresh” function for a set of sheets.
- A possibility to retain a drawing after changing a sheet.
- Standard drawing functions like a free drawing tool, lines, circles, rectangles, polygons etc.
- Standard text-functions with the ability to change size and colour of text.
- A clear function.

In Figure 23 is the visual appearance given for an example of such an HCI view (note: the file-option is missing in this version of the mock-up).

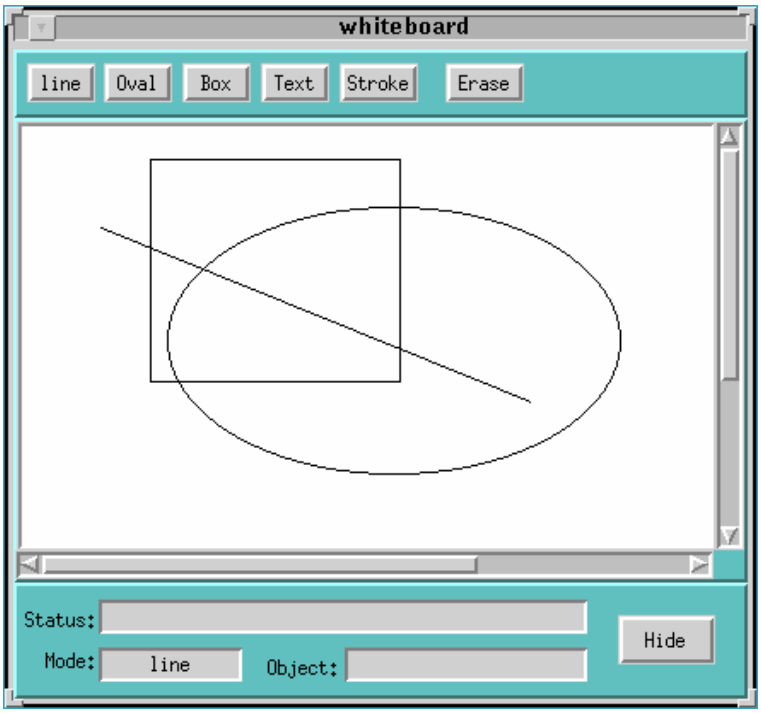


Figure 23 Whiteboard HCI-view mock-up.

4.7 Floor management

4.7.1 Introduction

This chapter mostly discussed, until now, the HCI design for the DTC system. The DTC system consists, however out of more parts than just the HCI.

The first part consists out of an already existing session-layer implementation (see [2]). This takes care of transport of both media-data and control-data over the same transport-layer and presents audio and video directly to the user. The latter has been done out of efficiency requirements and is recommended in some standards (see [12]). This part has been coloured light grey in Figure 24.

The second part consists out of the HCI. The design of this HCI has been described in the sections before this one of this same chapter. This HCI processes events from the session-layer and gives commands to that same session-layer. Those commands and events can concern control-data or the control of the presentation of direct audio/video. This part has been coloured white in Figure 24.

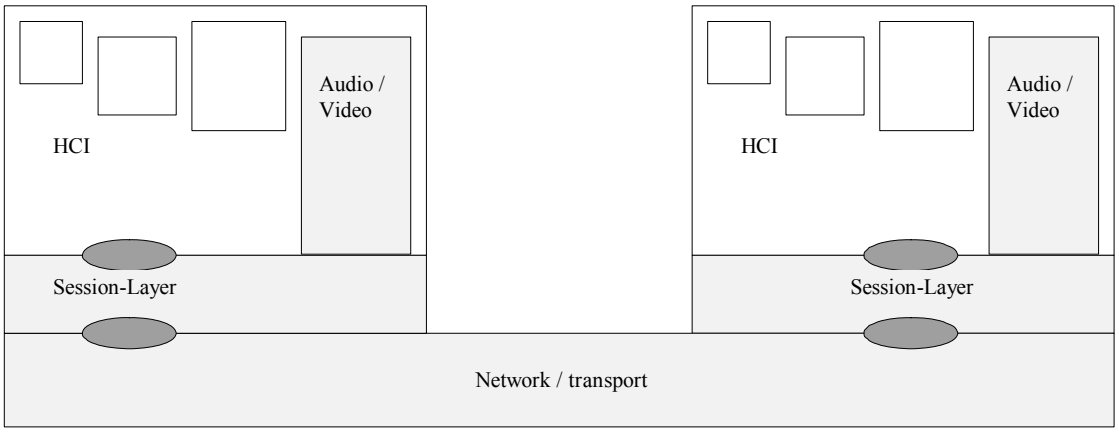


Figure 24: The Relation between the HCI and the session-layer within the DTC-system.

4.7.2 An extra floor management layer

As mentioned in Chapter 2, the session-layer does not provide complete functionality to manage floors. Only the service-primitives to exchange floor-control information and to influence media-streams are available. Nonetheless, floor-information has to be kept and enforced somewhere. To do this, an extra sub-layer must be introduced.

There are basically two options to design such a floor management layer. The first is to design it as part of the session-layer and the second is to design it as part of the HCI-environment.

The main advantage of implementing the floor manage layer within the session layer would be a more strict enforcement of floor management, independent of the HCI that is used.

The main disadvantage would be that the session-layer has hardly any knowledge about roles or floors. Therefore, floor management is not easy to implement within the session layer itself. A floor-database and role-database has to be maintained. The service-primitives would have to be extended to incorporate floor definition and role-definition.

Within the HCI, this information about roles and floors must already be present to be able to uses these floors in the first place.

Therefore the choice was made to extend the HCI with an extra layer, the Floor Management (FM) layer.

Strictly speaking, this is therefore not only a HCI anymore. In the remainder of the text the whole layer above the session-layer will be referred to as the application-layer.
The position of this layer is illustrated in Figure 25.

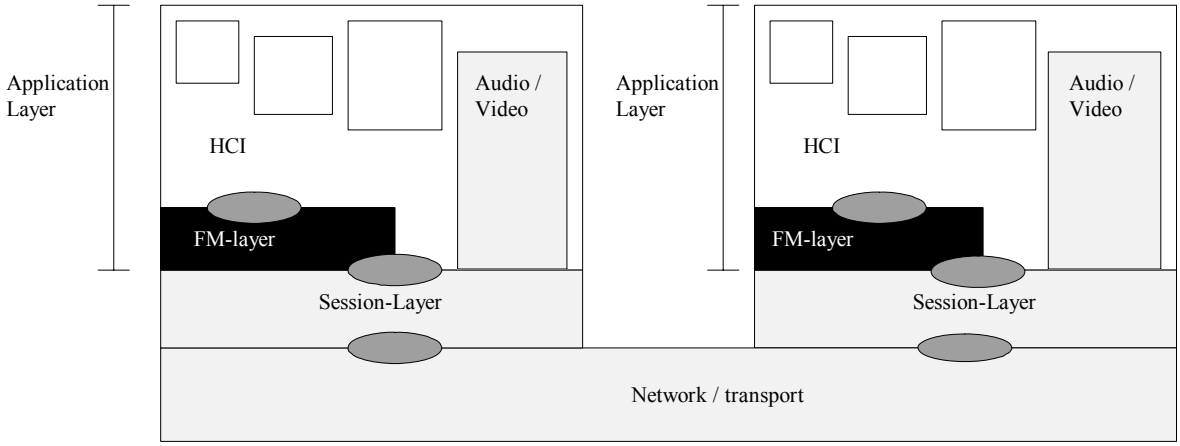


Figure 25: An extra Floor management (FM) layer has been added.

4.7.3 Floor information

To manage all floors, the floor management layer must take care of the properties and the definition for each floor. This incorporates read/write permissions and the role of the participant for that floor, as described in the E-R models from Section 3.

There are a number of additional demands that must be met. First, floors must have a unique identifier so it is always possible to refer unambiguously to a certain floor. Second, since the session layer uses network (IP)-addresses to identify users and direct all PDU's, the address of the Floor Controller has to be stored, so requests can be properly addressed. This could also be done by storing user-id's and use a mapping later, but we have chosen to use network-addresses because this layer is closer to the network-layer and the mapping to user-id's will only take place at HCI-level. Floor-type and media channel indications are needed because an incoming data-stream or a given HCI-command must be mapped and applied to the right floor by the floor control layer.

Because of all these demands, it means that the floor control layer must keep a database in which for each floor the following items are being stored:

- Floor ID.
- Source permission (Read Token).
- Sink permission (Write Token).
- User-role (participant or controller).
- Floor controller address (This is needed when a user is Floor Participant: address to direct requests to).
- Floor-type.
- Media-channel.

From a HCI-point of view, the following items are also important and will be kept in the database:

- Floor write muted (to determine whether or not to actually start transmission).
- Floor read muted (to determine whether or not to actually start reception).
- Floor name.

Floor read- and write-muted are needed so a floor participant or controller can control his own access to a particular floor, in combination with the permitted access-rights.
A floor name is necessary because a floor needs, besides a unique identification (ID), also a "friendly" name to be presented to the user.

When the DTC-system starts, this information is being read from a number of files. This is later described in the description of the implementation in Chapter 5. There is as yet no means of dynamically changing floor definitions, since the session-layer does not provide service primitives for that purpose.

4.7.4 The management of floors

Each time a floor participant wants to read or write from a certain floor, this must be checked by the floor management layer. When the necessary tokens are present, the operation will be performed.

In case of direct media streams (audio or video), reading or writing is started and from then continues on its own. At the start of the transmission, the access properties can be checked, but not during the rest of the media-transfer. This means that whenever the appropriate access permissions are taken away by the controller or given back by the user, that floor has to be stopped automatically by the floor management layer. This is illustrated in Figure 30 - Figure 32. Stopping of a transmission can occur always, since there is no permission needed.

It was also decided that, whenever a read or write permission is given, reception or transmission for that floor is automatically started, when permitted by the local user. This has the advantage that a controller can actually remotely switch floor access on and off. This is illustrated in Figure 29.

Since the floor management provider knows the address of each floor controller and the session layer does not, this has to be filled in during a floor-request, floor-request-withdraw or a floor-return. This is illustrated in Figure 26 - Figure 28.

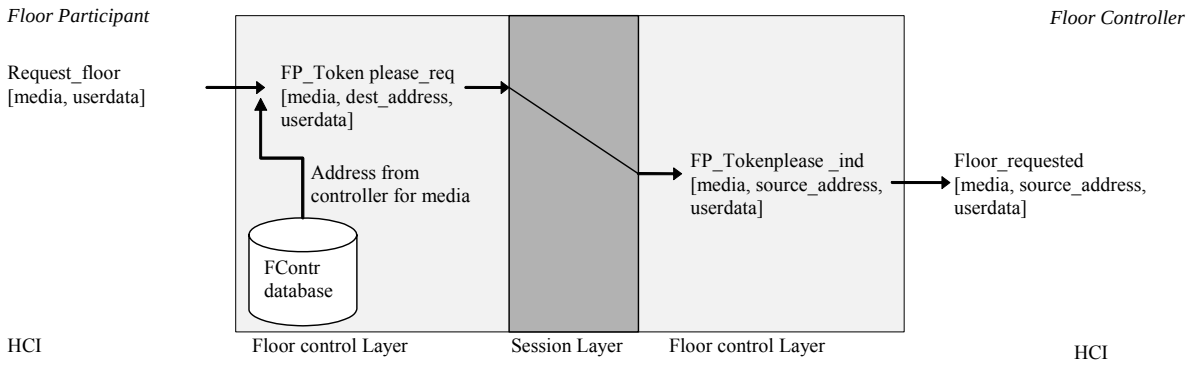


Figure 26: Requesting access to the floor.

The floor control layer has to supply the right address for the floor controller.

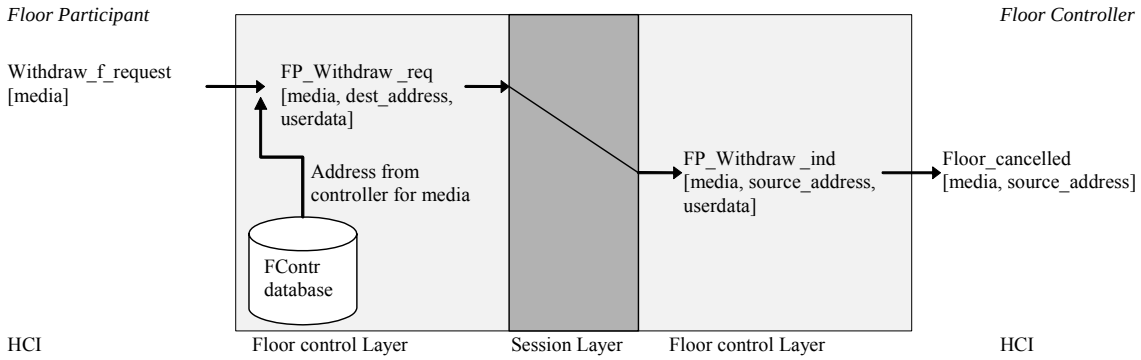


Figure 27: Withdraw a request for access to a floor.

The floor control layer has to supply the right address for the floor controller.

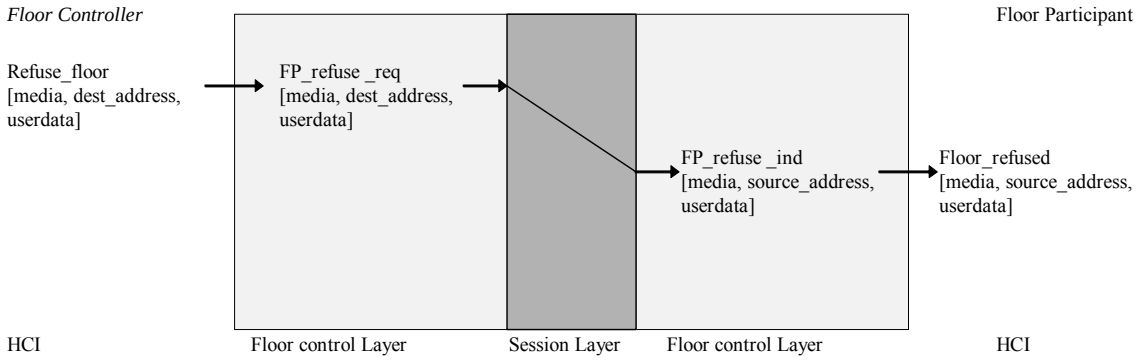


Figure 28: Refuse a request for access to a floor.

This is just passed through.

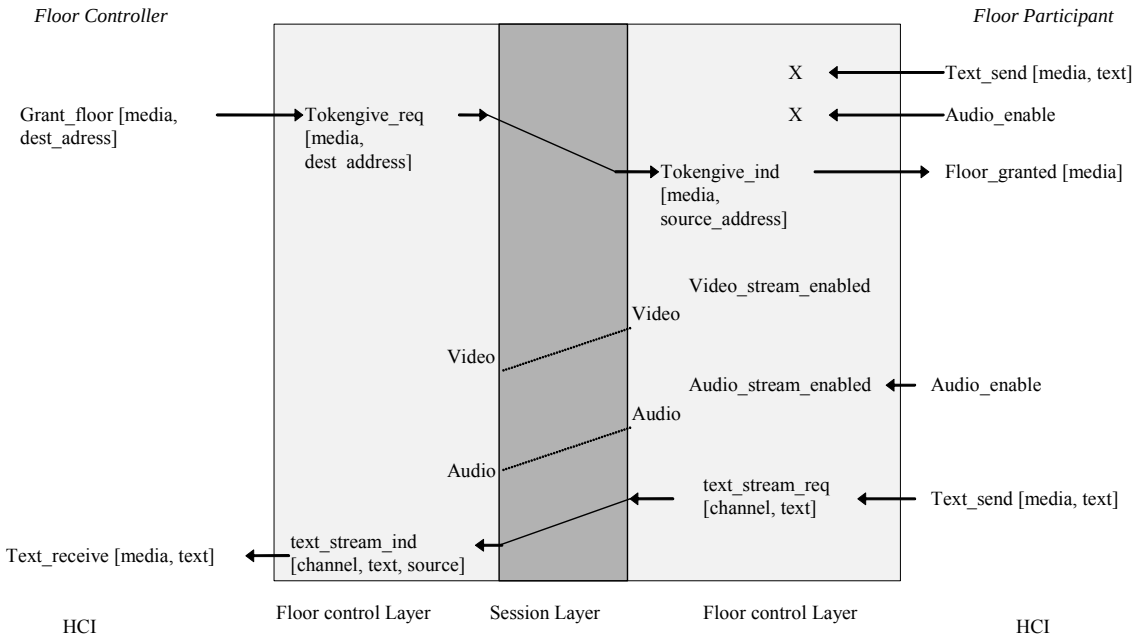


Figure 29: Granting write access to floors.

This is an example of a grant for writing on a video- , an audio- and a text- floor.
Assumed is that, in the beginning, no write access was permitted. Therefore write instructions to one of the floors are ignored.
After the grant, writes to the granted floors will be started automatically at the side of the participant, or, dependent of media- type and -state, can be initiated by the HCI.

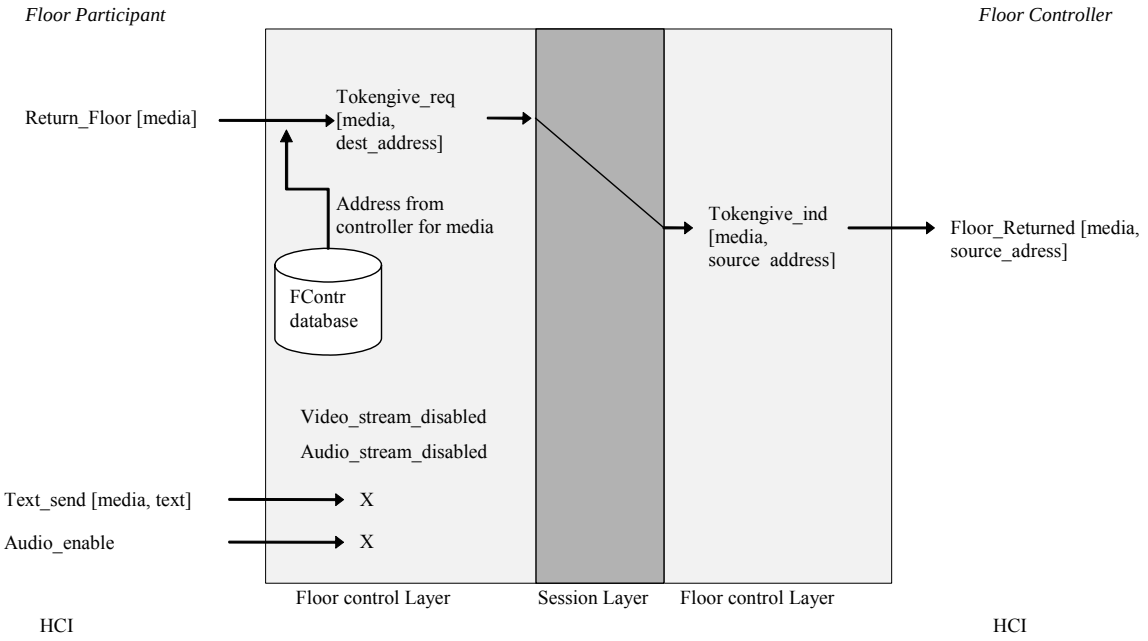


Figure 30: returning write access for floors.

This is an example of a return for writing on a video-, an audio- and a text- floor. The floor control layer has to supply the right address for the floor controller. After the return occurred, writes to the granted floors will be stopped automatically at the side of the floor participant. Write instructions to one of the floors are ignored.

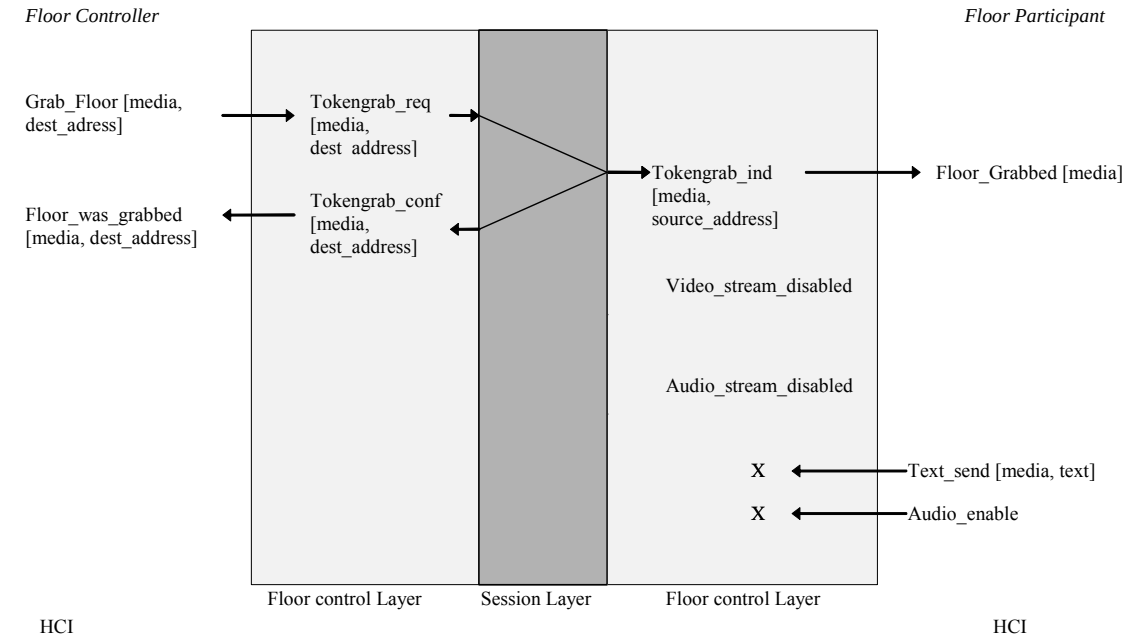


Figure 31: Grabbing write access for floors.

This is an example of a grab for writing on a video- , an audio- and a text- floor. After the grab indication occurred, a confirmation will automatically be given to the initiating side by the session-layer and writes to the granted floors will be stopped automatically by the floor control layer. Write instructions to one of the floors are ignored.

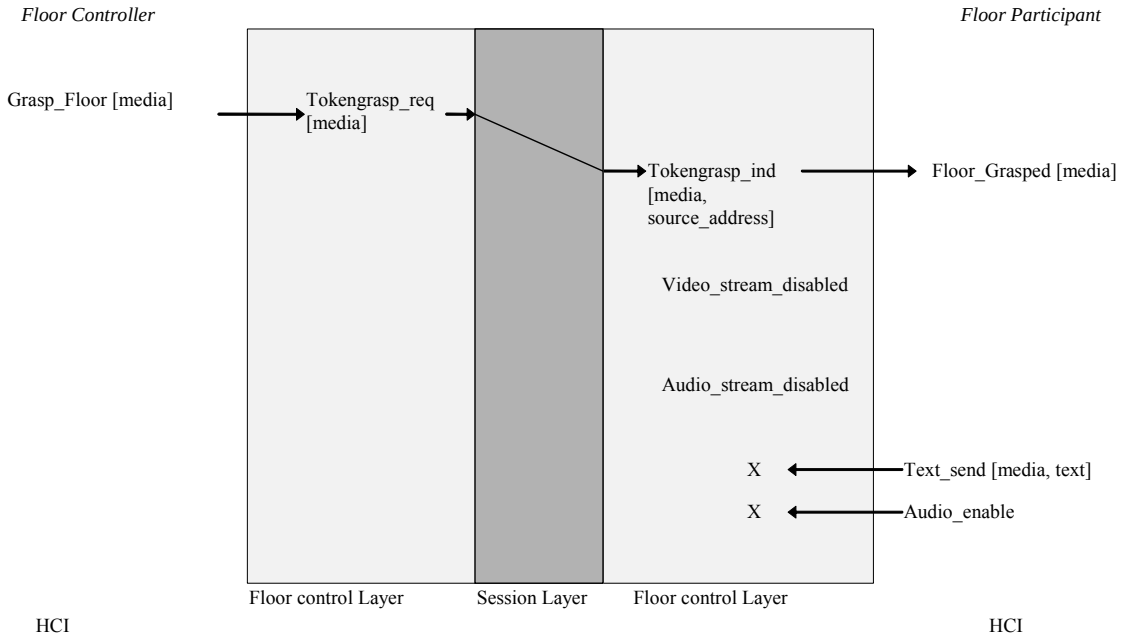


Figure 32: Grasping floors.

A Grasp Floor command acts just like a Grab Floor command, but now it grabs the floor for all participants.

5 Implementation

5.1 Introduction

This chapter describes the implementation from the Desktop Tele-classroom Conference system. This implementation makes use of a number of requirements as given in chapter 2 and the HCI-design as given in chapter 4. It describes the subdivision of the DTC-system in layers and modules and the function of each separately.

5.2 Implementation requirements

A number of requirements, as described in chapter 2.5, were kept in mind during the implementation of the HCI for the DTC system.

First of all, the HCI and the floor-management layer of the DTC-system both had to be implemented on a Sun Sparc workstation, using an already existing session-layer.
Second, TCL/TK and Guibuilder were already chosen as a development-environment for the HCI. Because of the advantages of hard/software independence, also the floor-management layer of the DTC-system itself was implemented using this already chosen environment.
Last of all, since this system might be working together with administrative systems, user-information has to be kept in simple, separate, databases-files.

5.3 Main architecture of the DTC system

5.3.1 Division into layers

The DTC application has been divided into a number of layers. It was decided that such a division was necessary since this DTC consists of different parts that each can be developed further and have be able to use different hard- and software.
This layering therefore makes a large part of the code more or less independent of the other parts. Hereafter the layers will be summarised.

- The HCI layer is at the top and contains the actual HCI. This layer provides the HCI-views and takes care of all data-management, floor definitions and floor use at HCI level.
- The floor management layer controls floor access. The need for such a separate layer and its design has already been described in section 4.7.
- The session independent layer provides a translation between the actual session-layer and the floor management layer.
- The OS independent layer provides a translation between the TCL/TK code and all used external programs and scripts. This layer is therefore is directly dependent on the used Operating System and hardware.
- The external programs and scripts. The session-layer already exists. The rest consist of small support-programs and scripts to control audio and to improve stability of the DTC system.

See also Figure 33.

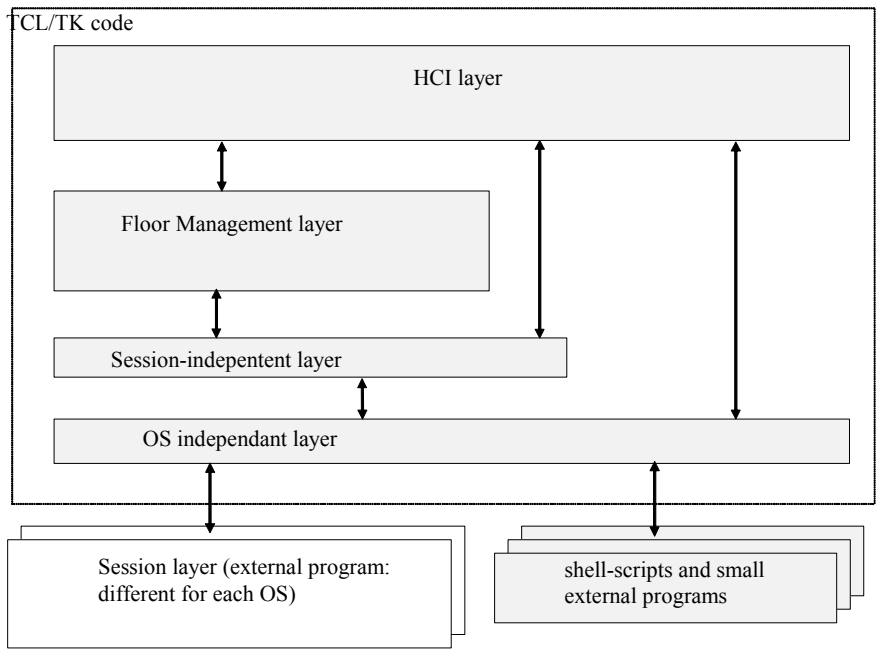


Figure 33: Division into layers.

The grey modules are part of this design.

5.3.2 Subdivision into modules

The system is subdivided into a number of modules. These modules each perform a number of coherent tasks or control a certain amount of coherent data. The layers as described above are made up of one or more modules.

These modules are mere conceptual clusters of procedures, used within the graphical design-tool "Guibuilder" (see appendix B) by grouping them in main-window-like templates. The actual scope of global variables and of procedure-definitions is not influenced by those modules, but the modules are constructed in such a way that global variables for data or status-information are only used within one module and not shared between modules.
This module division has been done because TCL/TK, the language that is used for this implementation, provides no object oriented facilities. These could have been added as an extension that was available, but that would have rendered the adapted version of TCL/TK without B&O-support.

5.4 The architecture of the layers

5.4.1 The HCI layer

Main HCI-view

The first number of modules is derived by using the HCI-views, as defined in Chapter 4
These are already described in that chapter and consist of 6 main HCI-views and are directly used to define modules. These consist of a windows definition for that HCI-view, together with all procedures that act directly upon it.

- start-window (2 versions: a real and a test-version)
- Floor-management window
- Group-management window (mock-up)
- Local Media control window.
- Text-media window (there can be multiple instances of this window-type)
- Shared whiteboard media window (mock-up)

The start-window has two versions that are implemented. One for testing purposes and one is for use within the actual DTC system. The test-version disables the use of the databases that contain many fixed options. Therefore many parameters can be dynamically adjusted.

External HCI: Pictures of faces

Provisions have been made to use this application in combination with another TCL/TK application made by Michel Schudel [14]. This application shows pictures (faces) of meeting participants.

HCI popup-window

Furthermore, there are a number of popup windows. Those are not main HCI-views or windows, but small windows that give or ask for information in more detail. They also have their own procedures.

These windows are:

- Floor-request with reason window
- Floor-reject with reason window
- Floor-information window

HCI Debug-window

There is also an optional debug/control-window present that shows all communication between the session-layer and the Floorcontrol-layer and gives the opportunity to issue TCL and session-layer commands. This is meant for debugging purposes, both for the development of the HCI itself and for future development of the session-layer. It can be started in both start-up modes, but is automatically enabled in the test mode.

Initialisation

To initialise the whole application, an initialisation module, called Main, exists. This parses the command-line parameters and gives control to the right start-window or, if enough parameters are given, can even start a DTC-session directly. This last option can be used to start a DTC-session from within an automated environment.

The hierarchical coherence between the different HCI-view and -popup modules is shown in Figure 34.

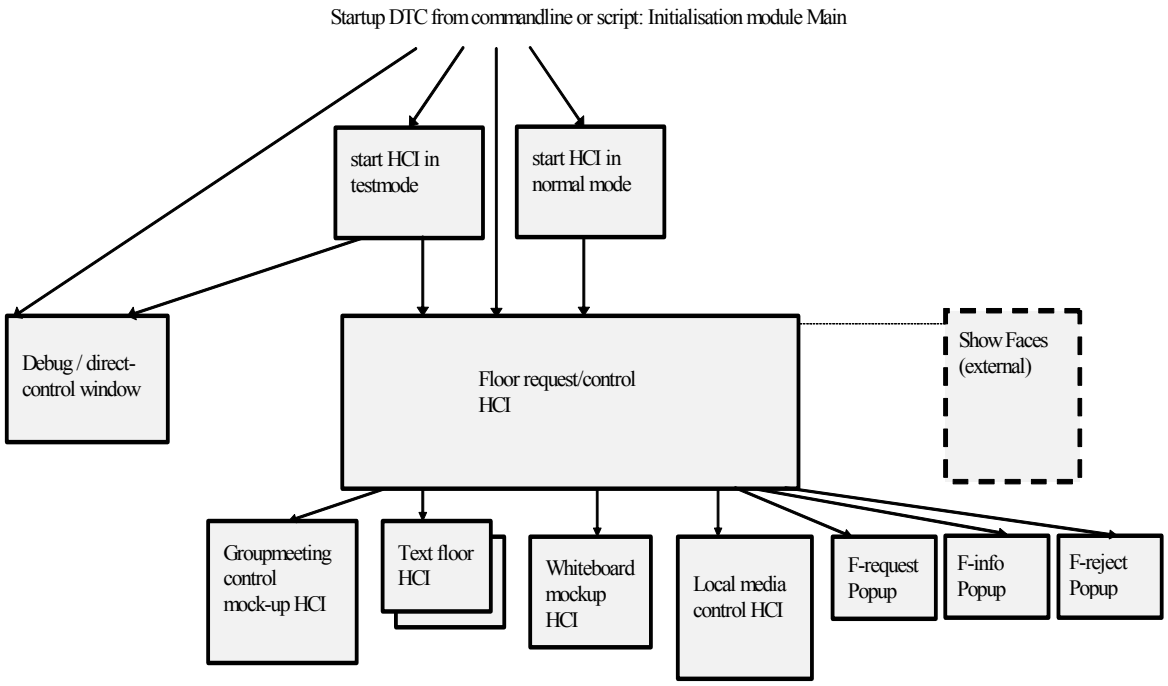


Figure 34: Hierarchy of HC-view modules
The arrows indicate the hierarchy that is present within the modules. The start-HCI shows the Floor request/control HCI and optional a debug /direct control HCI. The Floor request/control HCI is used to start all other HCI modules. The command line interpreter can also bypass the start windows and begin a DTC-session directly.

Data and definition management

There is a substantial amount of data that has to be managed within the DTC-HCI and there are predefined lists with definitions of names, DTC-meetings and roles in those meetings for the DTC to be used. To do this, a number of modules are needed to read those definitions and to maintain data-structures that are used within the HCI. This has been done in separate modules to make the actual information, as kept by the HCI, independent of the presentation in the HCI-views.

- The user-data module keeps track of users. It reads the users names from the user-name list and it registers their user-numbers and network-addresses and maps user-names to user-numbers. To perform this mapping, this module uses an external database file. The HCI can use this to map incoming and outgoing floor-events to users.

- The request/grants module keeps track of all requests and all grants for both the Floor Controller and the Floor Participant.

- The floor database module reads the names and reflector addresses of all meetings for the user to be able to select a meeting and join it. When the user joins a groupmeeting, it reads the groupmeeting definition databases and extracts the needed floor information. Next, it defines all needed floors with all necessary parameters for a given combination of user and groupmeeting. Furthermore, this module keeps track of the roles for all others users. This is used by the HCI, for instance, when a new user joins the groupmeeting, to find out whether this user is a Floor Controller or not.

Floors / media usage

To define and use floors in a generic way, an extra module is needed to take care of the visual appearance of floors and, when needed, their accompanying local media controls. This is called the floor-windows module. It defines and creates the right type of window for each floor-instance, when needed, and it takes care of floor status-messages.

5.4.2 The floor management layer

The need for such a separate layer and its design were described in section 4.7. In short, its function is to provide means to control floor access.
This layer was not implemented within one module, but has been divided into three smaller modules to keep it more organised. These modules are shown in Figure 35.

- The floor-definition module maintains the definitions for all floors as explained in 4.7.3. Within this module all information of floors at floor access and floor management level is kept. It maintains both static and dynamic floor information for all floors.

- The Floor-use module controls the use of individual floors. This incorporates reading/writing, checking read/write permissions with the floor definition module and the actual data-streams or enable/disable functions for each floor.

- The floor-request/control module processes floor requests, grants returns and grabs. It checks for floor-control permissions with the floor-definition module when necessary and passes information both to the main HCI and the floor-use module. It also processes the adding and deleting of Floor Participants.

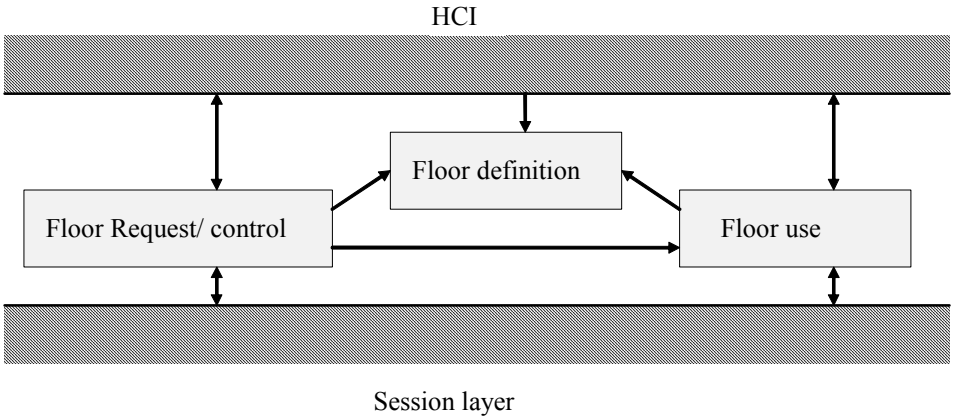


Figure 35: The floor-management layer.

The lines between the modules and layers stand for "usage". When a module uses another module, it makes calls to procedures within that module.

5.4.3 Session-independent-layer

Because the session layer implementation was still in development, the choice was made to concentrate all calls to and responses from the session-layer in a separate layer that was put in a separate module. This module can also handle some problems with the session layer implementation that are still present. Most important of that is the handling of control commands for video-streams within the session-layer. Access to the real session layer implementation all goes through this layer. This layer, in its turn, uses the OS-independent layer described below.

5.4.4 Operating-System Independent layer

This layer provides a translation between the TCL/TK code and all used external programs and scripts. This layer is therefore is directly dependent on the used Operating System and hardware.

TCL/TK in itself is Operating system (OS) –independent, but OS-independence for the DTC-system as a whole was not possible since a number of non TCL/TK programs and scripts are part of the DTC system. Therefore, the choice was made to concentrate all OS-dependant parts of the DTC-system in a separate layer that takes care of conversions and chooses the right script/program for the required function, depending on the OS that is being used. This layer

consists of one module. All calls that involve direct contact with non-TCL/TK programs are being executed by procedures within this module.

This OS-dependence is amongst others the case for the session layer implementation, since this is one of the non-TCL/TK programs. As for the different Operating Systems: Solaris 2.4 and Windows NT 4.0 are being supported by different versions of the session layer (The Solaris session layer implementation was made by Alex Jongman [2] and the windows NT session layer implementation was made by Marc Berenschot [13] within this implementation. At this moment, only the Solaris version has the ability to transmit video.

5.5 The modules

5.5.1 The initialisation module (Main)

The procedure “main” in main-module is called when the TCL/TK script is started. This parses the command-line parameters and then transfers control to the chosen start-window with the extracted parameters. If the parameters determine it, those start-windows can also continue directly with the DTC session, without waiting for user-interaction. This can be used to start the DTC system in an automated fashion. For a description of the command-line parameters: see Appendix J.

5.5.2 The start-up HCI-view

This screen exists in two versions. A test-version, where all parameters must be filled in, and a normal, login-version

Test Version

This version accepts a number of parameters and uses them to start the actual groupmeeting, without using the database-files. It limits the roles to two pre-defined ones: Teacher and Student.

The parameters that can be changed are:

- Name
- Reflector address
- Role (Teacher or Student)
- Video Floor-access controlled by teacher or free for all
- Video-compression (CellB, Jpeg, UYVY or Mpeg1)
- Debug window on or off
- Sound full- or half-duplex
- Use IP-over-ATM

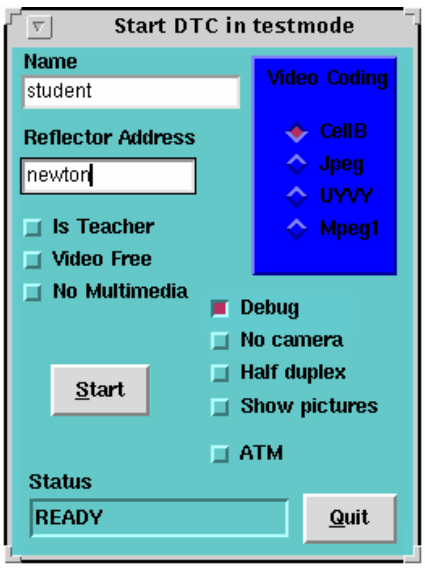


Figure 36: Start-up HCI-view for the test mode.

It has a function to automatically switch the basic name when the role is being changed. It also has a function to calculate the IP-over-ATM-address belonging to a normal IP-address and vice-versa.

The main procedure from this module is w_st1:start_conference that is called when the start-button is pressed. This procedure takes all the adjusted parameters that are globally present within this module and tries to join or start a conference. For a more thorough description, see the next section about the login version.
.

Login version

This version is the start-up HCI-view that is presented to the normal user.
It accepts only a user-number or -name and a meeting-number. The meeting number can also be chosen from a dropdown-list.

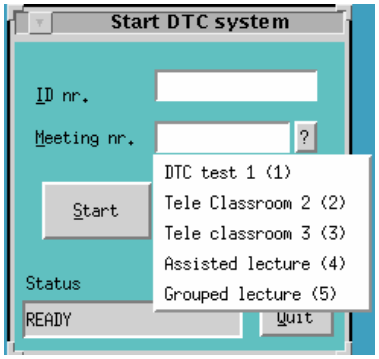


Figure 37: Start-up HCI-view for normal mode with drop-down list.

The main procedure from this module is w_st2:start_conference that is called when the start-button is pressed. This procedure takes all the adjusted parameters that are globally present within this module and tries to join or start a conference.

First, it completes/converts the username. Then it defines some local-media parameters. After that, the network layer and the floor request/control HCI are initialised. Then, the floorcontrol layer is defined and initialised. After that, the floor request/control HCI is shown, the start-up-HCI is hidden the network layer reception is enabled. Before that, all incoming messages were temporary stored within the network layer until they could be processed.

The sequence [network layer initialisation + floor HCI initialisation] -> [floor definition] -> [floor HCI show + network layer enable] is needed because they are interdependent. The floor definition needs part of the floor HCI to be defined in order to start giving correct status-information and the network layer is needed to start transfer control-information, but no information may be received from both HCI and the network layer until all definitions are in place and consistent.

5.5.3 The floor management HCI-view.

This module has many procedures that deal with the HCI-part of floorcontrol functions and call the appropriate HCI sub module procedures. They call the appropriate functions in the floorcontrol-layer or are being called by the floorcontrol-layer upon events.
It has a main HCI module and 4 distinct sub modules. The main HCI module handles the general state of the HCI and most of the communication with other modules.

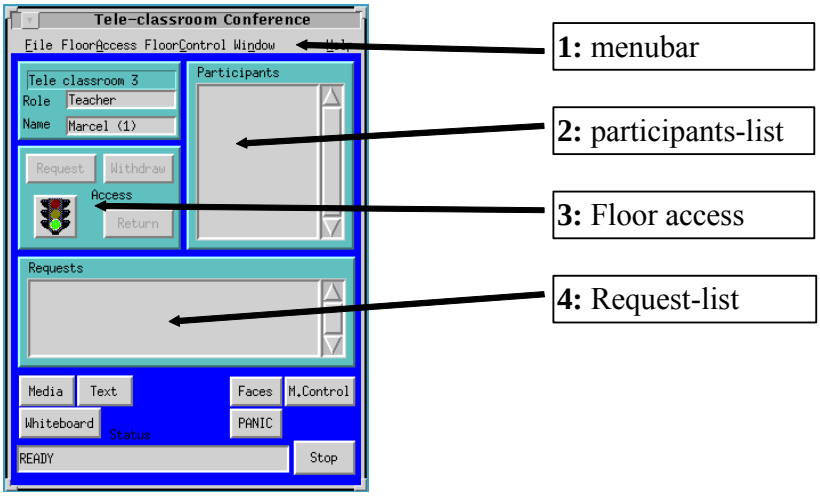


Figure 38: floor management HCI with HCI-areas managed by sub modules.

The first sub module handles the menu bar. It handles menu-actions invoked by the user. It keeps track of which menu-items are possible. It dynamically creates menu-items that are needed separately for each groupmeeting participant. This is needed for actions like requesting/enabling access to a floor (see Figure 39). The second handles the groupmeeting-participants list. Visually adds or deletes participants (see Figure 40). The third sub module handles floor access status information, requests, grants, withdrawals, and returns. It visually shows the access status to all floors. In this module, the next design cycle suggestion for a combination of floor controller and participant both controlling and requesting floors has already been implemented (see Figure 40). The fourth sub module handles the floor-access request list and shows all pending requests in the order they arrive (see Figure 40).

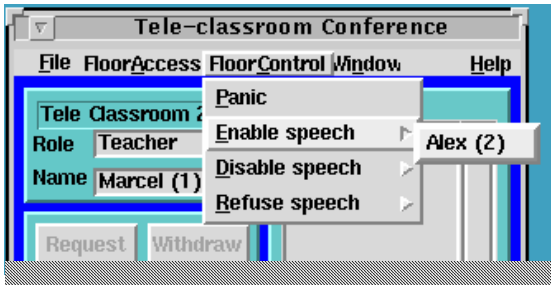


Figure 39: Example of an unfolded menu with one user added.

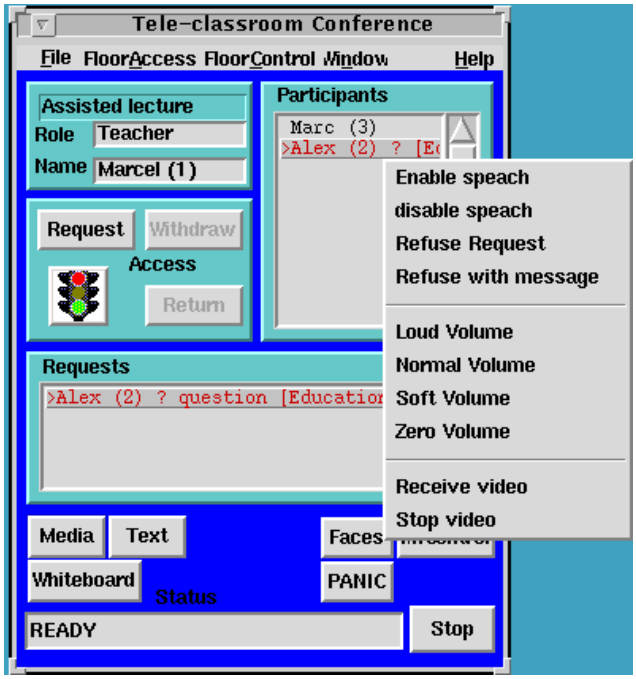


Figure 40: Example of a pull down menu in list with users.

Two users were present on the list and one has a pending request. This is combined with an example of the current user both being floor controller and floor participant.

5.5.4 The floor status HCI-view

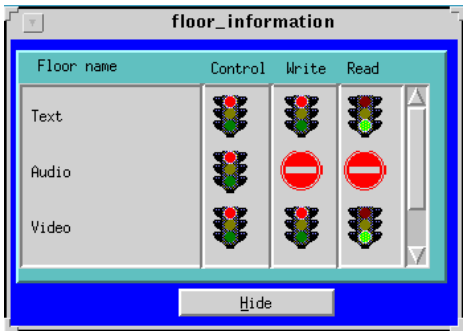


Figure 41: Floor information HCI with a medium that's disabled.

This HCI-view shows the status information for all known floors within the groupmeeting. It is updated by the main floorcontrol HCI each time when something occurs about floor access.

5.5.5 The local media control HCI-view.

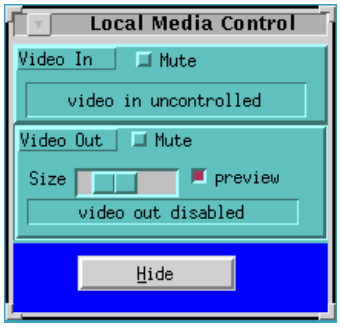


Figure 42: Local media control HCI with some parts hidden.

This HCI-view controls the local media. Those are audio and video. It lets the user change media-properties within allowed ranges. It also lets a user mute/un-mute any local medium. Each block is a small sub module that is only visible when the accompanying medium and direction is actually usable.

5.5.6 The text-floor HCI-view

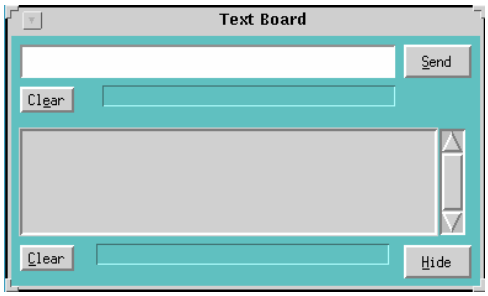


Figure 43: Text-floor HCI-view.

This is HCI-view for a text-floor. This window can have multiple instances that are numbered like the text-channels. Each of its procedures also has that channel as a parameter.
Because this window can have multiple instances, bindings for buttons can't be created at design time, but must be created in runtime during the initialisation of a new window-instance to incorporate the right window number / channel.

5.5.7 Refuse request popup.

Figure 44: Refuse Request popup.

This popup HCI gives the user the possibility to specify a refusal-message instead of just refusing a request.

5.5.8 Floor request popup

Figure 45: Floor request popup HCI.

This popup-window is meant to request floors. It contains a message-field to give a request-reason and a floor pull down-list to select the floors to request.
This pull down-list is filled with all single floors, with all different combinations of 2 floors controlled by one controller and with an entry that contains all floors. This makes heavy use of the floor definition module.

5.5.9 Whiteboard mock-up HCI

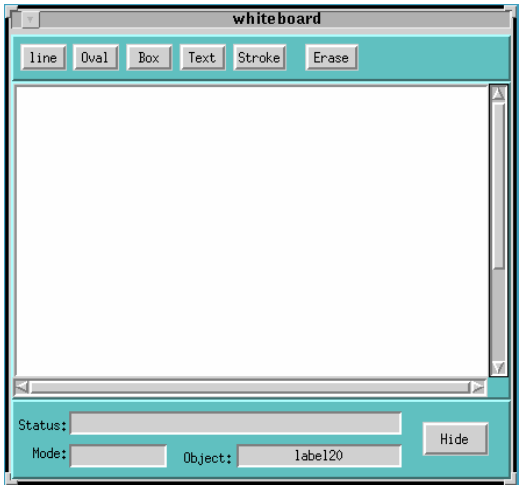


Figure 46: Whiteboard mock-up HCI.

The whiteboard mock-up HCI has no real floor functionality, but it can be used to draw simple figures and might in the future be adapted for real use.
The canvas-code is derived from an example by Brent Welch [11].
It uses a status-variable to decide what kind of figure must be draw. While drawing, the figure is constantly being deleted and drawn again, until the mouse button is released.

5.5.10 The groupmeeting control mock-up HCI

This has no real functionality, only a “hide” function.

5.5.11 External module: pictures of faces.

Provisions have been made to use this application in combination with another TCL/TK application made by Michel Schudel [14]. This application shows pictures (faces) of meeting participants, when available. When appropriate, calls to the external interface of that module have been inserted in the HCI. It shows black and white pictures of all defined participants for a groupmeeting and shows the ones of users that joined the groupmeeting in colour (see Figure 47).

It also indicates floor sink rights where appropriate. The database this applications uses, is, at the moment, not the same as the one used by the DTC.

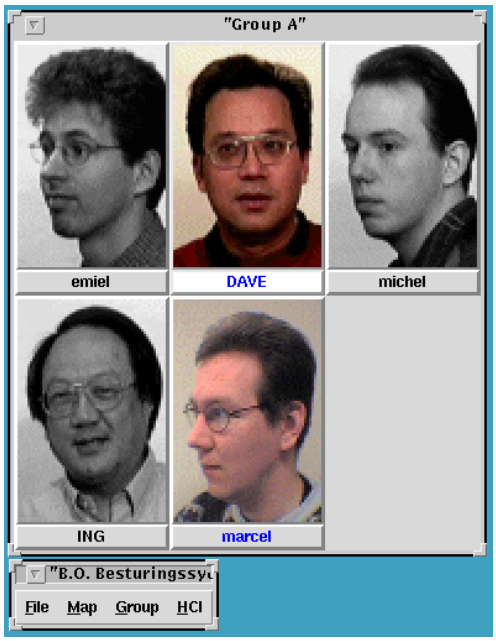


Figure 47: External module with pictures of faces of participants.

5.5.12 The user database module

The user database module keeps track of all static user-information and of the network-location of users. It registers their user-numbers and network-addresses and maps user-names to user-numbers. To perform the mapping from user numbers to usernames, this module uses an external database file “tele_u.ini”. The HCI can use this to map incoming and outgoing floor-events to users.

5.5.13 Request / grants module

The request/grants module keeps track of all requests and all grants for both the Floor Controller and the Floor Participant. Requests and grants are stored in a number of module-global arrays/lists and can be stored or retrieved on basis of their network-address.
The Floor Controller can use this module to find out which floors were requested by whom to grant or refuse them. Also, the Floor Controller can use this module to find out which floors were granted to whom in order to grab those floors back from the concerning Floor Participants.

The Floor Participant uses this module to keep information about requested floors in order to be able to cancel a request for all requested floors. In addition, this module will be used to keep track of granted floors in order to be able to return them to the concerning Floor Controller.

5.5.14 The floor database module

The floor database module initially defines the roles for all users.
It reads the groupmeeting-definition databases. This module reads the groupmeeting-name database “tele_mn.ini”, the groupmeeting-template database “tele_mt.ini” and the actual role mapping-database “tele_m.ini” that maps roles from meeting-templates onto user-id’s. It extracts the needed floor information and takes care of the definition of all needed floors with all necessary parameters for a given combination of user and groupmeeting. This is done within the floor definition module of the floorcontrol-layer. Furthermore, this module keeps track of the roles for all others users. This is used by the HCI, for instance, when a new user joins the groupmeeting, to find out whether this user is a Floor Controller or not.

5.5.15 The floor-windows module

The floor-windows module (floor_hci) takes care of the visual appearance and instantiation of floor HCI’s and, when needed, their accompanying local media controls. It defines the right type of window for each floor-instance, when needed, and it takes care of floor-status messages. Here it is also possible to indicate that some local media functions are not possible.

5.5.16 The floor control module

This module handles all incoming and outgoing floor-requests and calls the appropriate floor definition functions, floor use functions, network layer functions and HCI-functions. This module does not contain any status-information itself.

5.5.17 The floor use module

This module handles all incoming and outgoing floor-information and passes it or blocks it, according to the information in the floor definitions module. It calls the appropriate floor definition functions, network layer functions and HCI-functions. This module does not contain any status-information itself.

5.5.18 The floor definition module

This module maintains the definitions and status for all floors. Within this module, all floors are created or destroyed at floor management level. The floor-information is stored in a number of global named arrays and can be changed and retrieved by a number of procedures/functions.

5.5.19 The network layer module

This module takes care of any conversions and control needed for message between TCL and the actual C++ network session layer implementation. Parameter lists are concatenated or split up. Incoming messages are being parsed here to call the appropriate floor control or floor use procedures. This layer also takes care of the serialisation of all local-media commands that are sent to the C++ session-layer because a flood of commands that is too fast can lead to a crash of the C++ program. Furthermore, this layer takes care of the video floor reception, if present. This is needed because, for a video-floor, there might exist more than one video source. Each of these can be enabled or disabled at will. We want to provide the user with a “mute-all” and “enable all” function. This module, therefore, keeps track of all requested video-sources. Video sources can be added or deleted, all muted or all enabled.

5.5.20 System independent layer module

This module takes care of the actual interface with the operating system and the C++ session layer and shields the rest of the DTC-system for different pipe-handling methods, process and memory management. It is aware of the current operating system and performs system-calls accordingly.

5.5.21 Summary

In Figure 48 the connection between al the modules is being displayed.

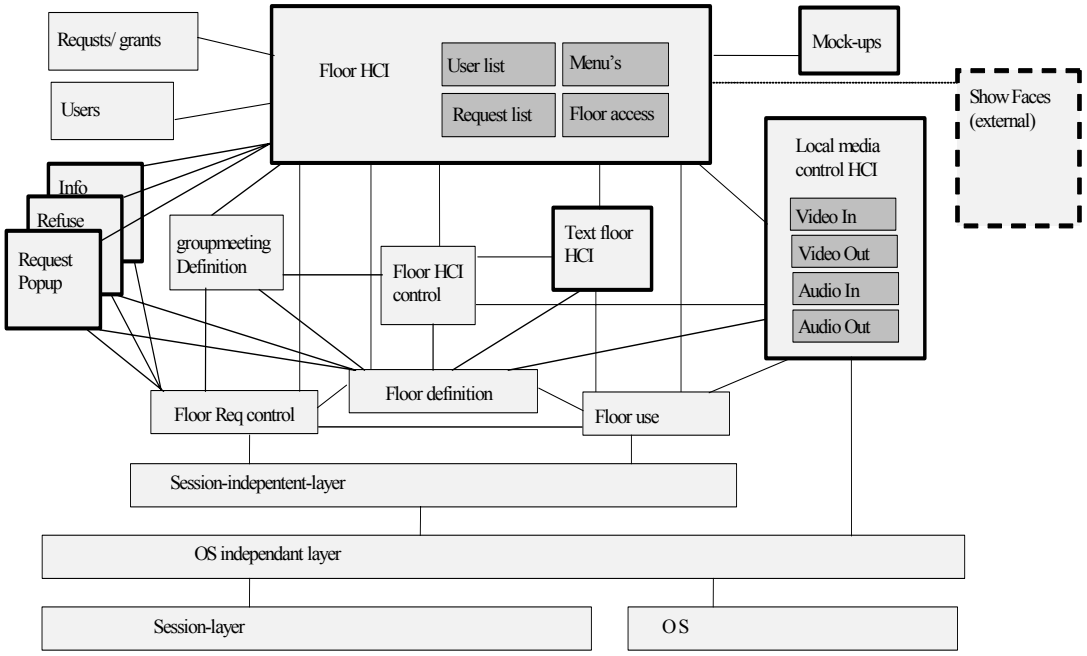


Figure 48: Connections between the different modules.

This gives the connections between the different modules while the HCI is running. The Staring and stopping has been left out since, in that case, every module needs to be initialised or reset. All HCI-views have a thicker borderline.

5.6 Implementation details

The complete listing of the DTC HCI implementation can be found in Appendix O. A shorter listing of only the procedure headers, partly with descriptions, can be found in Appendix L.

5.6.1 Databases

A number of suggestions coming from an administrative education application [5], described in appendix B were used for the DTC system. These concern the division of information over a number of different small databases and are explained in further detail in Appendix H

The databases used are:

- A user-ID database, mapping numbers to names
- A groupmeeting ID database mapping numbers to names
- A Groupmeeting definition database mapping users to actors and setting a meeting-template
- A groupmeeting-template database, defining all actor-types, floor types and teleconferencing-roles

5.6.2 Interfacing with the network layer

The HCI will not directly call functions from the session-layer since this is written in C and the HCI in TCL/TK.

It might be possible to compile the session-layer into the TCL/TK interpreter. This would greatly enhance performance, but that would reduce the compatibility with future systems since this has to be done again for each new version of TCL/TK, that is still in development. Furthermore, it would reduce the ability to test the session-layer independently. This layer is also still in development.

Therefore, the choice was made to let the HCI start the session-layer as a completely separated process and use a Unix-pipe-mechanism to communicate in both directions. This works also for Windows NT, with minor modifications.

5.6.3 Session-layer problems

The session layer implementation was unstable under circumstances. This was because only one video-command could be processed at a time. In Appendix I the measure are explained to prevent this from happening.

5.6.4 Video-windows and the colour palette

There are now 200 colours used by the video window(s). That leaves a rather small palette for the rest of Xwindows and the HCI of the DTC application. There are no colours left for new graphic enhancements, like more icon-symbols. Furthermore, new additions like a whiteboard will be very limited in the use of colours.

5.6.5 Design-Tool

The design tool that was being used for TCL/TK, GuibuilderV1.0, has been adapted to suit the growing needs during the implementation. A description of Guibuilder and the needed adaptations is given in Appendix K.

6 Trials

There were not really held any structured trials yet, but I noted some comments from people who experimented with the DTC-system.

Some noted drawbacks were:

- Some of the terms used in the DTC system are not really clear.
- There are almost no icons or pictograms to make understanding easier.
- The colour-differences in the floorcontrol-list between green and red are not always to clear and the letters are small.
- The HCI Application was sometimes sluggish and the video window is small or too slow.
- Audio-quality is not very good and is often not comprehensible.

7 Intermediate conclusions and recommendations

7.1 HCI

Visual

The HCI looks good, but does not provide enough visual attention to some events and some terms like “floor” are not always clear.

In future versions of the DTC system, it could prove better to provide the buttons with icons instead of texts. This might be more intuitive for inexperienced users.

The coloured lists do not really stand out. The use of images with front- and side-views, as implemented by Michel Schudel [14] is a good idea to attract attention from users, but the request-subjects are not shown there, so a teacher still needs to look at the lists.

Functional

The picture-add-on made by Michel Schudel [14] took the conference a level higher to a full conference with possibilities for more than one groupmeeting. This means that more media of the same kind like audio would be coexisting in one conference. The session layer implementations are at this moment not yet capable of this.

With the service primitives as present in the current session-layer implementations, in order to use them, a floor participant must already know the identity of the floor controller for each floor. This can only be done with the groupmeeting-templates, but it can’t be changed dynamically. It would a good extension to incorporate this in future session layers.

It might be possible to implement a simple whiteboard without adapting the session-layer by using a modified text-channel.

Speed

It might be possible to implement the session layer as a TCL-function in a recompiled TCL/TK interpreter. This will probably be a lot more efficient.

Stability

There are still some stability-problems when using video.

7.2 Details about the problems with the current session layer and how to solve these

Colour palette

There are now 200 colours used by the video window(s). That leaves a rather small palette for the rest of Xwindows and the HCI of DTC application. A video-window should be able to display adequately with only 128 colours, leaving more for other applications. Alternatively, a high- or true-colour display could be used.

Video functionality

CellB does not work with larger images (network layer crashes). That should be looked into. Mpeg does not work at all between Suns. This should also be corrected because Mpeg gives a better image quality and is a more universal standard.

Performance is still not good for larger images. This might be solved by better hardware.
It is technically possible, but not implemented yet, to change the brightness of a video image (Sun's Showview can do that). It could be very useful to correct a too dark or too light image.

Video-network layer stability

The TCL/TK network layer does a lot of checking to ensure video commands are serialized to prevent a crash of the network layer. This would be far more efficient and secure when implemented within the network layer itself. An alternative is to let the network layer use another internal system to transfer commands instead of named pipes.

Still, it is not completely stable. The SUN-version that can exchange an Mpeg1-stream with a windows-NT version is not stable when used with CellB.

8 Extra design cycle

8.1 Aim

Due to circumstances, this assignment was suspended in 1998 although a lot of work was already done and a working prototype of the above HCI was already made. Now, in 2007, this assignment is allowed to be continued and to be finished. Meanwhile a lot of time has passed while the IT has developed enormously. This means that a part of an extra design cycle, using current knowledge and techniques, is required to actualize this assignment. The remainder of this report aims to do the above.

8.2 Organisation of this second part of the report

This report exists basically out of 2 parts. The first part, consisting of the previous chapters (Chapter 0 to Chapter 7), treated the initial assignment. The second part, consisting of this and the following chapters (Chapter 8 to Chapter 13), treats the actualization of this assignment in the form of a part of an extra design cycle for the HCI of the DTC system.

This chapter, Chapter 8, introduces the actualization of this assignment.

In Chapter 9, a new literature study will be introduced to actualize this assignment. At the end of that chapter, new constraints for the design of an improved DTC system, with emphasis on the HCI of such a system, using current knowledge and technologies will be given.

In Chapter 10, the information from Chapter 9 will be combined with the existing user requirements from Chapter 3 to arrive at new user requirements for such an improved DTC system.

In Chapter 11, the above user requirements, together with the new constraints as given in Chapter 9, will be used to give a number of recommendations for a new HCI design. Since this is done as an actualisation of an assignment, it will not result in a complete finished design. In addition, a number of recommendations will be given for improvement of a selected current teleconference system.

Chapter 12 gives an elaboration on a few design and implementation details, selected from topics described in Chapter 11. It will not result in a design for a full working prototype of a new DTC, but will allow the exploration in some detail of some modern techniques and theories.

Chapter 13 contains the final conclusion and recommendations of this report.

8.3 Background

8.3.1 Scope of time

The first part of this report is about 10 years old. Developments in the IT-world are proceeding rapidly. During the past period, hardware as well as software has improved considerately. Therefore, much of what has been done in this first part may be outdated in some or other way.

8.3.2 Progression in hardware

There are numeral advancements at hardware level. The most relevant ones with respect to teleconferencing are listed below.

Interfacing

- Graphical capacities of displays and display-adapters have increased resulting in a persistent use of 24 bits true-colour on desktops, combined with higher resolutions like 1280x1024 pixels and larger size screens like 19”and 21” that are available on the consumer market [15]. Therefore, issues with colour palettes as mentioned in Section 7.2 do not occur anymore. Due to those higher resolutions, a GUI design now can have more details.

- 10 years ago, digital multimedia technology was still in an early stage. Most digital multimedia devices like cameras were expensive and not common. Nowadays it has developed into maturity. Current video- and photo cameras are mostly digital. These can directly interface with computers [45] [46] [47]. This means that it is now possible to demand that, for using a DTC application, workstations all need to be equipped with video cameras.

- Laptops have progressed to a level that laptops are now widely sold and are for many an affordable alternative for workstations [52]. Handheld devices (now many are smartphones) have made a big development and are now increasingly common [53] [54]. This means that a new version of a DTC-system may not only be used on static heavy workstations set in a (class) room anymore, but may now also used in less controlled environments on laptops and even on small mobile devices.

Networking

- Networks using wires have become faster and cheaper. During the first design phase of the DTC system, 10 Mbit/s was normal. At this moment 100 Mbit/s is normal and widely available [43]. Connection to the internet is now done mostly by means like ADSL and Cable (The speed ranges from 256 Kbit/s to 20 Mbit/s [43]). All of those are fast enough for good quality video transmission. The old DTC system had an option to directly use an ATM network to transfer high quality video that nowadays is not necessary anymore.

- Networking these days can also be done wireless. Several working wireless networking standards, like WLAN, GPRS, UMTS and Bluetooth are now widely available [43] providing speeds op to 100 Mbit. Together with cheap wired networks (see above), this creates possibilities for ubiquitous networking. This provides users with an internet-connection

regardless from where they are. Combined with handheld devices it provides users with a large freedom of movement. A DTC system useable on wireless phones could be particularly interesting in a number of third world countries that do not have a large internet infrastructure yet, but already have a good cell phone infrastructure in place.

Processing

- Today, chip sets are designed to perform 3D operations in high quality directly in display adapters [42]. This results in many real-time high quality 3D applications. This could open the possibility to design the DTC system as a 3D virtual classroom.

- Workstations have become much more powerful and cheaper. As Moore's Law is still in effect [41], both processor capacity and memory increased enormously with about a factor of 30 during the last 10 years. Next to being faster, current processors are equipped with specialized instructions to accelerate multimedia operations [55]. This means that it is hardly necessary anymore to look for methods to improve the speed of a DTC application and/or its programming environment, as mentioned in Section 7.1.

Storage

- Storage has an enormous increased capacity. Hard disks available at the consumer market nowadays can have a capacity of 500GB [44]. This results in the possibility of storing and retrieving large amounts of high quality video in real time on computer systems (both servers and workstations). Combined with the above-mentioned advancements in multimedia technology, this means that multimedia is mostly available in digital form these days. It is therefore not necessary to be able to use analogue sources of audio / video signals, like in the old DTC system, to include recordings. Instead, it should be possible to include digital pre-recorded material stored on the local workstation or on a server in a DTC-session. In addition, it could now be possible to record (parts of) a DTC-session to replay this later.

8.3.3 Progression in software

Compared to 10 years ago, software support for networked applications and networked applications development has progressed for a fair amount. Some examples are given below.

- In software development, almost every programming language now available has extensive developing tools assisting in the creation of an interactive HCI. To support this, a number of general API's have been developed to design and implement real-time HCI's and other system functions. These API's are general and not bound in particular to a specific programming language. An example of this is .NET [15] (and it's Linux-counterpart MONO [39]). During the first design cycle of the DTC system, it was difficult to find a suitable development system for its HCI (see Appendix B) and even then, it was necessary to combine systems and adapt them, to be usable enough. At this moment, there are a number of integrated development systems to choose from to develop a new DTC HCI.

- Ubiquitous networks (see Section 8.3.2) enable a development from workstation-based software towards online software using e.g. AJAX (Asynchronous JavaScript And XML) [22]. Toolkits for designing applications or for converting them from normal Java are now being developed. This might also be applicable for a new DTC design.

- Advanced real-time video compression and decompression technologies are now available in most operating systems. This has been done in the form of new standards [18] [50], resulting in standard codec's and players. Examples are the Windows Media Player with codec's [49] and the multi-platform system Videolan [38]. MPEG4 can even be used to model a 3D environment using VRML-like definitions and to navigate in such an environment [18]. The used algorithms result in either the use of relatively low bandwidth for normal quality real-time video transmission or the use of normal bandwidth for very high quality real-time video transmission (for example WMV9 [50] can do both). A newly design DTC-system should make use of those possibilities.

- There is a growing use of specialised, extendable open source frameworks. In the course of time, a lot those frameworks have been developed with different purposes. One example is a web-based content management system called Joomla [51]. This can be used to design and maintain a coherent series of dynamic web pages. It might be possible to find a suitable framework, designed for educational purposes and to use it as enveloping environment to integrate the DTC system into.

8.3.4 Advancement in HCI theories and practices

During the past 10 years, knowledge about HCI design has been progressing. This has resulted in improvements of which some important ones for the design of a DTC system are:

- Better specification methods. A number of specification methods like state charts [15] have been developed that are supported by software tools [56].

- More and better design principles and guidelines have been developed during time [15].

- More knowledge about using collaborative software is now available. In [15], [16] and [17] about the same division is given for collaboration systems dividing it in space and time. The DTC system falls in the same time / different place quarter. Knowledge about HCI design for this type of collaboration is therefore of most interest.

	Same time	**Different time**
Same place	Synchronous local, face to face (meeting rooms)	Asynchronous local (e.g. team scheduling, group calendars)
Different place	Synchronous distributed (e.g. chat, instant messaging, video / audio conference)	Asynchronous distributed (e.g. e-mail, newsgroups, listservers, discussion boards, blogs, wiki's)

Table 3: Division of communication in space and time.

- Next to old interaction styles like command line, menus, direct manipulation [15], new interaction styles like virtual reality and augmented reality [15][17] are now advancing.

- Advanced usage of interaction devices like image recognition and speech recognition is more and more possible [15] [19] [20].

8.3.5 Conclusion

All of the above technological advancements give a strong indication that the design of a state-of-the–art DTC system from 10 years is outdated. Furthermore, research of Human Computer Interaction has advanced because much more is possible using present computer systems, which typically are connected, by diverse network topologies.

This provides the challenge to investigate possibilities to improve the DTC-system as designed and implemented in the first part of this report with current knowledge and technologies. In the next section, an approach for this improvement will be given.

8.4 Approach

8.4.1 General strategy selection

To perform such an improvement of the old DTC application as mentioned in the previous chapter, at least a part of an extra design cycle is needed. Several strategies to take on such a design cycle are available of which 3 will be further explored below:

Strategy 1: Updating the old application

A first strategy is to update the existing DTC-application described in Chapter 5. This can be done by adapting the HCI-part of the program to use versions of the programming environment and tools as described in Section 2.4.2 that will run in a current environment. Furthermore, the implementation of the session layer as described in Section 2.4.2 must be adapted to run in a current environment. This would result in a version of the old DTC system that works on present computer systems. This can then be improved by updating the external “look” to reflect the look and feel of today’s applications to some extend.

Current workstations at the University of Twente run Solaris 8 or Windows XP. A small attempt was undertaken (see Appendices I, J and K) to get the old DTC system working again. The HCI part is now working on current workstations. Unfortunately, to get the DTC-application actually to work, the session layer implementation (as described in Section 2.4.2) would have to be redesigned. This is necessary because it does not work on anything but Solaris 5, Windows NT 3.51 or Windows NT 4 (see Appendix I). Regrettably, the source code is not available for several parts of this implementation.

The drawbacks of this strategy are, that it does hardly add anything extra to the already existing system, that it does not use any real new technology since it uses “old” tools, and that it is very improbable that it is possible to rewrite the non-working session-layer in the available time. At the end it is still very well possible it will not result in a fully working prototype.

Strategy 2: New Implementation based on the original requirements

A second strategy is to take only the original user-requirements of such a system as stated in Chapter 3 and start from there. When needed, these requirements could first be adapted to the current knowledge about HCI design and to the current state of technology. Next, the design and implementation of the DTC system can be started. This will be based on the before mentioned adapted original requirements and will be done using present technologies.

Strategy 3: Start from existing teleconference applications

A third strategy is to acknowledge that, during the past 10 years, working teleconference applications have been created that satisfy a number of modern HCI design requirements and start from these.

These applications already incorporate a number of the requirements for a DTC system as stated in Chapter 3. First, the most suitable application must be selected. Next, by including the missing requirements for a DTC system, this can be used to arrive at a new DTC application using this current teleconference application as a basis. This can be done by giving design recommendations to extend the functionality of that application. In addition, research about meeting techniques and research about new interaction methods could be incorporated. This can be used to arrive at a number of extra design recommendations to enhance it even further. Next, maybe some concepts could be demonstrated by making a small prototype showing some of those enhancements.

A major drawback of this approach is that most current fully working teleconference systems are not extendible since they are not open source. Therefore, those specifications can only be derived by observing the system and or its user-manuals. Those that are Open Source are still in an early phase and not quite finished. This makes it difficult to retrieve exact specifications to determine missing requirements. In addition, it is hardly possible to include a prototype to show some enhancements in an existing system.

Strategy: Conclusion

The second strategy is chosen to let the work in the following chapters be closely connected to the work in the previous chapters while being able to research and use modern technology. It is combined with the third strategy to profit from the fact that existing applications have progressed much. This means that the old DTC requirements will be used and adapted where needed. In addition, current, already existing teleconference systems will be taken into account, as will research into present meeting techniques and interaction methods. This will result in design recommendations for a new DTC application and in recommendations to improve one selected existing teleconference application. When possible, a few concepts mentioned in the above recommendations will be implemented in small prototypes.

8.4.2 Literature study

A new literature study is needed; incorporating current theories and practices for HCI design, information about new interaction methods, present technologies, and about current programming tools.

8.4.3 New requirements

The requirements of the old DTC-system will be analyzed from the perspective of current theories about HCI design and knowledge about present technologies. This will result in new, improved requirements for a new DTC-system.

8.4.4 A more generic design

It could be argued that a new design for a DTC-system should be more generic, but that seems hardly necessary for the following reasons:

- The design of the old DTC-system was already based on requirements for a more generic teleconference system. This was done by mapping the specific functions of a DTC-system on that of a more generic teleconference system (see Section 3.5).

- During the first design cycle, only the parts needed for the core DTC application were designed, but during a small second design cycle in Section 4.6, some mock-up user interfaces for more generic usage were already designed and implemented.

- More parts of the old DTC system were already designed and implemented, using a more generic approach. The text floor GUI was already made capable of a more generic usage by being able to handle text floors with different floor controllers. In addition, the underlying floor control layer is designed on a more generic basis (see Section 4.7). The configuration database file definitions are also able to handle more generic teleconference setups (see Appendix H).

The focus of the second part of this report lies more on the usage of new theories and technologies, this part of the report will not concern itself much with aspects of a more generic design.

8.4.5 A design using current technology

The new requirements mentioned in Section 8.4.3, combined with the knowledge gained as mentioned in Section 8.4.2 will be used to give a number of design suggestions for a new DTC-system. Some of those recommendations will be explored deeper, resulting in one or more some small prototypes demonstrating current techniques and HCI design for a new DTC system.

Since these prototypes are only small and will probably not be carried further by other students, the above is only feasible by using software the university does not have to buy for this project. In addition, software that is very complicated to install and /or maintain should be avoided.

9 Current literature study

9.1 Approach

This chapter provides on overview of current theories, practices and technologies that can be used to improve the DTC system.

As stated in Section 8.4.2, this includes the following:

- Current theories and practices for HCI design. These will be taken from some learning books about HCI design.
- Information about current and new interaction methods. This will be taken from some recent research at the HMI department of the Twente University and from the IEEE catalogue.
- Programming environments and tools. Some articles comparing different tools / environments / frameworks will be used to select a few. These will be reviewed more closely.
- Existing modern teleconference systems will be reviewed by looking at the manual of a few of those. Real testing might have been better, but it is mostly impossible to set them up just for testing. This because it might require licences and\or complicated server setups.

These will be presented in the next sections.

9.2 Current theories and practices for HCI design

To design a HCI, a number of elements are needed. First, the design process as a whole needs to be considered. Theories and models on HCI-design might help the first design steps. Knowledge about collaboration mechanisms can help to refine the design. Specification methods are needed to specify the most important interaction aspects of a HCI. Design principles on how to design a user interface further guide the design. The design can be refined with more specific and concrete design guidelines for HCI's. These guidelines can be even more refined with extra design guidelines for collaboration software.

9.2.1 The HCI design process

In [15] an outline is given on how organize the HCI design process. This can best be based on three important elements:

- Usage of guideline documents. Early during the design process one starts with the creation of guideline documents. These documents start with a set of working guidelines based on theories and models about HCI design. Those can also contain subjects about design decision policies, education of users, etc. These documents are continuously maintained and adapted during the rest of the design process.
- Usage of good software tools to design and implement user interfaces, both for the finished product and for prototyping.
- Expert reviews and/or usability testing. Expert reviews are done by co-experts. Usability testing is done by controlled experiments with a group of test users for evaluation.

The old DTC system design did not use many guidelines, but the design conformed to most of the standard Xwindows conventions at the time as for placement of menus and shapes of widgets. The software tools (Tcl/TK and GuiBuilder were quite advanced for those days and performed well. Unfortunately, controlled experiments on groups of test users were never performed due to circumstances.

9.2.2 Models and theories about user interface design

A number of theories and models are presented in [15], but as also is explained in [15], most of those have a very limited use. Even the most useful model presented, the Object-Action Interface Model (OAI), has limited use according to [15]. The reason for this is that it is rather trivial when applied to direct manipulation style HCI's that is used by many current HCI's. This style is also the style of the HCI of a DTC system (see Section 9.3.1).

9.2.3 Specification methods for HCI's

To specify Human Computer Interaction, a number of modelling and specification methods exist. In [15] a number of modelling and specification tools are treated like grammars, menu-selection trees and dialog-box trees, transition diagrams and state charts.

Grammars

Grammars are descriptions of the syntax of commands. They are meant for command line interfaces. These are not very suitable for a modern interactive user interface according to [15] since they have difficulty coping with 2D information styles, like most modern user interfaces today have.

Menu-selection trees and dialog-box trees

Menu-selection trees and dialog-box trees are tree-structures mapping all possible choices that can be made in a menu or a dialog-box. They are quite powerful, and can show both high-level relationships and low-level details of a system and allow for consistency checks. A drawback of trees according to [15] is that they become difficult to handle with large systems. Another drawback is that modern direct manipulation interfaces do not only have menus to control the system.

Transition diagrams

A transition diagram for a HCI is a directed graph representing all possible states of a HCI and all possible state-transitions. It exists of a set of nodes that represent the system states and a set of directed links between the nodes that represent possible transitions. It is possible to label the nodes with transition frequencies when there are know. An example is given in Figure 49.

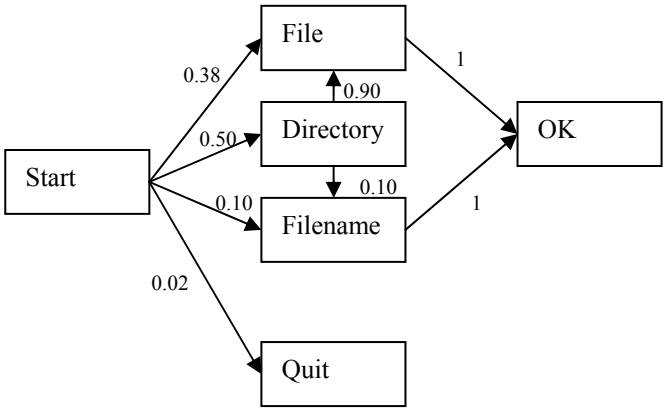


Figure 49: Transition diagram example for file manipulation actions.

Transition diagrams have the advantage that they can be converted almost automatically in finite-state automata and that several properties like reachability can be verified automatically [15]. Transition diagrams have the drawback that they are very difficult to handle when systems become more complex. This is because a standard transition diagram has no means for decomposition [15]. They also have difficulties to model concurrency and synchronisation [15].

Statecharts

Statecharts are an extended type of above mentioned transition diagrams. Statecharts do not have the above disadvantages of standard transition diagrams [15] [61] [62]. They have a grouping feature making it easier to handle complex systems and extensions are available to model concurrency, external interrupts, dataflow etc [15].

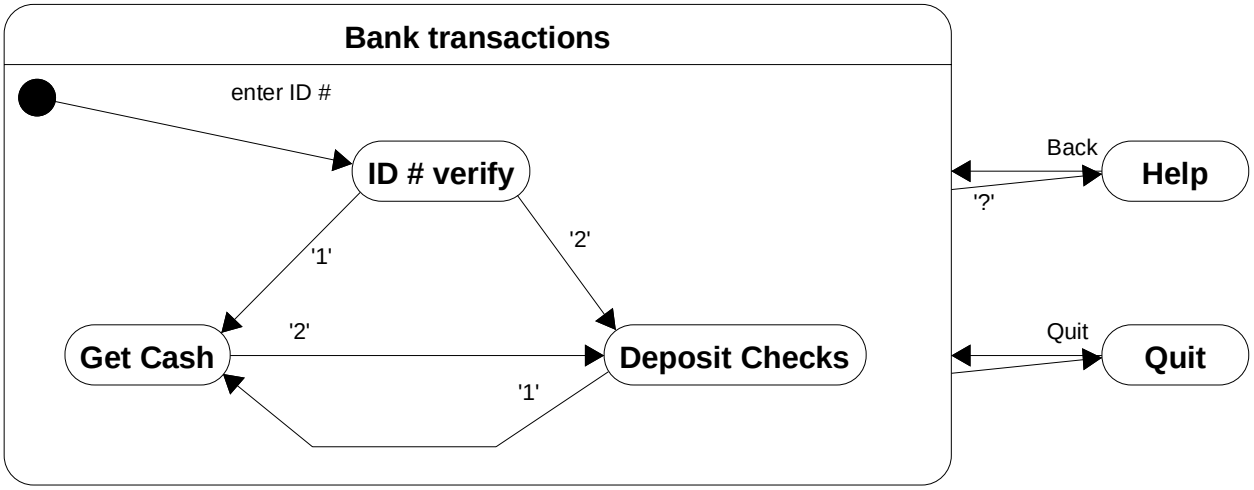


Figure 50: Example of a statechart for a simplified bank transaction system.

The Unified Modelling Language (UML) also uses state charts to specify the behaviour of general programs, so its support is growing and several tools are available to assist drawing state charts [15]. An example of such a tool is Tau 4.2 from Telelogic [56] that is available here at computers of the CS department at the University.

9.2.4 Design principles

In [15] many general principles, more or less general rules, on how to design a user interface are given. These can be subdivided into general HCI design principles and design principles for more specific HCI appearance and behaviour.

A selection of the most important ones with respect to designing a DTC system is shown below:

General HCI design principles

- Know the user: A designer must know for what kind of user the system will be designed. For instance, the targeted skill levels of potential users must be determined. Casual users need to find their way quickly without a steep learning curve. Experienced users will use the application intensively while probably first getting a thorough training. For instance, they can learn to use many keyboard shortcuts.

- Identify the tasks to be carried out and their frequencies: Identifying all tasks and possible sequences helps to specify a large part of the functionality of a user interface. It can also assist in deciding what tasks not to support to prevent a cluttered user interface that supplies an overload of tasks that users do not need. Furthermore, this can assist in determining what actions need to be atomic and what actions can be composites of other actions. For the old DTC system, the tasks were identified in Section 3.3.5, but frequencies of those tasks were unknown.

- Always provide some means of human control: Many routine tasks for a computer system will be increasingly simplified and automated. This helps users to avoid tedious and error prone tasks. Those automated tasks must have predictable and controllable interfaces that preserve human supervisory control. This is important since human judgement is sometimes needed to increase safety, to avoid failures and to increase quality. For a DTC system, it could for instance mean that it should always be possible for a groupmeeting controller to allow participants at will to a groupmeeting. Even if they have not subscribed in advance and the system therefore would not allow them to join automatically.

- Try to make a system's usage more universal: Implement for needs of different users and design the HCI so it is adaptable to those needs. The old DTC system was not much configurable as such. The location and size of many UI-elements were fixed and it was possible to use the keyboard by using menu-shortcuts, but not much direct shortcuts were defined (See section 4.5 and 5.5). However, the core of the DTC HCI was designed to be more universal. It was possible to adapt it to a number of more universal teleconference scenarios (see Section 4.7 and Appendix H) than only the DTC scenario.

HCI behaviour and appearance design principles

- Strive for consistency: E.g. always use the same key-combination for the same function, always place a button or menu-item with the same function at the same location, always give the same function the same name etc. This has been done in the old DTC system on many occasions. (e.g., status messages are mostly on the bottom of a window, the stop/hide button is always at the right bottom of a window). It could, however, have been done better (e.g. always use quit or stop, but not both, position the hide button always in the right bottom corner and not sometimes in the middle-bottom of a window, See section 4.5 and 5.5).

- Offer informative feedback: For every user action, there should be system feedback. For frequent and minor actions, the response can be modest, but for infrequent and major actions, a more substantial response should be given. The old DTC system always offered feedback through a status line at the bottom of the corresponding window and only serious events were presented as pop-up windows.

- Prevent errors: As much as possible, a system should be designed so users can't make serious errors. For example, menu-items that are not appropriate should be greyed out and users should not be able to enter alphanumerical characters in a numeric entry field. Most menu-items and buttons in the old DTC system were greyed out when not appropriate. Many selections were made from pull-down menus offering only possible selections. Only at the start-up screen, it was possible to enter wrong data, selecting a non-existent user or class.

- Permit easy reversal of actions: As much as possible, actions should be reversible. This relieves tension since users know errors can be undone. This also encourages trials-by-error to find out some unknown aspects of a system by the user since he/she knows mistakes can be undone. The DTC system itself permitted reversal of actions, where possible but sometimes reversal was not directly possible due to the nature of the system. In a number of cases, reversal is only possible with the cooperation of another user (e.g. e floor, once returned to the controller, can only be given back by that same controller).

- Reduce short-term memory load: An average person can only remember seven plus or minus 2 things for a short time. Displays should be kept simple; multiple page displays should be consolidated and, windows motion frequencies should be reduced. When not avoidable, the use of codes, mnemonics and sequences of actions requires the users to have enough training-time available and access to online help explaining those. The DTC system mostly used more or less complete names or, when numbers were needed for input, offered a list of choices extended with names. The biggest problem was that, when there were more floors defined, the number of windows became somewhat large to manage.

9.2.5 Design guidelines

Next to principles, there are numerous design guidelines. Guidelines are more specific compared to principles. They may not be the same for all types of interaction and may depend on the environment the system will be designed for.

In our case, for designing a DTC system, guidelines for organizing displays, organizing menus, using icons and dialog boxes are the most important ones. In [15], there are large lists of those guidelines. A selection with respect to relevance for designing a DTC system has been made.

A number of specific guidelines for organizing displays given in [15] are listed below:
- Present information in an efficient way to the user (make labels comprehensive, organize data in appropriate columns).
- Be consistent in labelling and standardize abbreviations.
- Use consistent formatting and graphic conventions.
- Present data only if it aids the user.
- Design displays in monochromatic form using spacing and arrangement for organisation and add colour afterwards only where it will aid the user.

- Minimize input actions needed by the user.

For usage of icons, some guidelines mentioned in [15] are:
- Limit the number of different icons.
- Make the icon stand out from the background.
- Ensure a single selected icon is clearly made visible when surrounded by unselected icons.
- Make all icons distinctive from each other.
- Ensure the harmoniousness of icons as a member of a family of icons.
- Use context sensitive help to explain functionality of icons.

Some guidelines for constructing menus are:
- Prefer broad-shallow to narrow-deep for a menu tree organisation.
- Use items as titles for sub trees.
- Group items meaningfully.
- Sequence items meaningfully.
- Use consistent grammar, layout and terminology.
- Provide keyboard shortcuts for frequently used items.

Some guidelines for the design of dialog boxes given in [15]:
- Give each form a meaningful title.
- Group and sequence fields in a logical way.
- Provide error correction for the user (e.g. backspace).
- Provide standard buttons (OK, Cancel).
- Keep the size small enough to reduce overlap problems; no overlap with required items.
- Display the dialog box close to related items.

Next to general HCI design guidelines, also, a number of guidelines may be used that represent the house style of the company that designs and/or the company that uses the system.

9.2.6 Designing for real-time different place collaboration

Social mechanisms

Social mechanisms are rules, procedures and etiquettes that have been established in our culture. Their function is to let people know how to behave in social groups. In communication and collaboration, those mechanisms are needed for successful collaboration [16]. Social mechanisms can be subdivided into three main categories: conversational, coordination and awareness mechanisms.

Conversational mechanisms are social mechanisms to facilitate the flow of talk. They enable us to coordinate a conversation, to detect and repair breakdown and to determine differences between formal and informal conversations. Most of these mechanisms work with voice communication [16].

Coordination mechanisms are social mechanisms to allow a group of people to act together. They are needed to keep meetings organised during collaboration [16]. These can be subdivided in 3 categories:

- Verbal (needs audio) and non-verbal (needs e.g. video) communications.
- Schedules, rules and conventions.
- shared external representations (e.g. checklists, shared calendars, document sharing)

Awareness mechanisms are social mechanisms to allow people in a group to find out what is happening in that group. This means: to find out what others are doing, who is talking to whom and to let others know what is happening. In a real-time different place collaborative system, these mechanisms can be very well supported by a lightweight chat application [16].

Problems in real-time different place collaborative systems might arise when people violate conventions of an online system out of "laziness". In addition, people tend to discard coordination mechanisms when they do not find them socially acceptable [16]. This means the right balance has to be found using system coordination by not giving the users too much freedom, but also by not forcing the users against their will.

Computer based communication mechanisms

All modern collaborative technologies like videoconferencing and instant messaging can be summarized as Computer Mediated Communication (CMC) [16] [17].

From earlier research into usage of CMC, it appears that people are flexible and adapt their communication style to the availability of different media [17] (an example is the use of emoticons in electronic text-only communication). This means that all available media in a DTC system can be useful and people can adapt their communication style to suit them.

Differences with non-CMC communication arise during group interaction, resulting in a more formal way of communication. For instance, during video mediated meetings, people adapt a more formal mechanism of turn-taking than during a real face-to-face meeting [17]. A DTC system supports these more formal turn-taking mechanisms by providing floor control.

The actual "presence" during a meeting is largely influenced by the richness of the available media [17]. Personal cues are better transferred using video signals that using text or audio-only channels. This poses problems for meetings that are more moving towards a virtual environment since it is difficult at present to detect and convey all social cues into such an environment [17].

9.3 Current and new developments in interaction

There are a number of interaction styles and devices for user interfaces [15]. Driven by faster hardware and new technologies a number of new interaction technologies have emerged. These vary from current ones that are used in normal production environments to state-of-the-art styles of interaction styles that are still being researched.

9.3.1 Interaction styles

Current interaction styles include the following: menu selection, form fill-ins, usage of command language, usage of natural language and 2D direct manipulation (manipulating or activating visual representations of objects or tasks on the computer screen). State-of-the-art interaction styles include 3D direct manipulation, tele-operation and 3D virtual reality.

Direct manipulation

A direct manipulation interaction style is an interaction style for HCI's that offers the users a natural, graphical representation of the possible objects and actions. Direct manipulation is still evolving and new variants are still being created. For now, three main varieties are known: 2D direct manipulation, 3D direct manipulation and tele-operation [15].

Direct manipulation has the advantage that it is transparent for many users. It makes the users feel they are involved directly with a world of objects and not with an intermediary. This is also shown when looking at the OAI-model in which the elements of the task block can be projected almost 1:1 on elements of the interface block [15]. It might still require users to have a substantial knowledge of the tasks, but once a user has acquired this, there is only a modest amount of interface knowledge required.

Drawbacks of direct manipulation are that it is more difficult for vision-impaired users to handle, that it consumes valuable screen-space, that it requires users to learn the meaning of visual representations and that those can be misleading. Users may under- or over estimate the functionality of those representations [15]. A last problem is that many direct manipulation interfaces frequently requires users to use a mouse and this might cost an experienced user more time that it takes to type in commands at a keyboard [15].

2D Direct manipulation

2D Direct manipulation is direct manipulation where all objects and actions are represented on the 2D surface of the screen: the desktop. It was the first of the direct manipulation styles to be developed. This is nowadays the standard user interface for many programs (Operating system shells, word processors, spreadsheets, games, Computer Aided Design (CAD), etc [15].

Many functions in a 2D direct manipulation interface are selectable by menus, but also by clicking on icons that symbolize those functions. Icons have the advantage that they can be smaller then corresponding buttons with text. This means they can be put nearer to the objects they are meant to manipulate, if applicable. At the same time, they can stand out more from the environment.

3D direct manipulation

Another, state-of-the-art variant of direct manipulation is that of a 3D interface. Some designers want to approach the richness of the 3D world by emulating this as exactly as possible in a 3D interface. This is then represented on a 2D screen. They believe that, the closer the interface resembles the real world, the easier usage will be. However, user studies [57] show that this is not always the case and it might lead to slowdowns because of increasing disorientating navigation, too complex user actions and annoying occlusions (An example of such a problematic 3D application is the usage of 3D desktops) [15]. For a number of applications like medical, architectural, and product design this is however still a strong option.

3D interface become a lot more powerful when they do not represent the actual world, but enhance it with features that are not normally possible like x-ray vision, teleportation, multiple views of objects etc. Many successful games are designed this way. Also in the above-mentioned applications, features like going back and forward in time, attaching labels to objects seem at least as important as a good representation of reality [15].

Tele operation

Tele operation is also a state-of-the-art variant of direct manipulation [15]. Tele operation means that a physical process is being controlled by a human operator at a distance (although the system may carry out some standard control tasks without interference). Designers must cope with problems like slower responses, incomplete feedback, increased likelihood of breakdowns etc.

Increased virtuality

The above-mentioned interaction methods were all based on reality, where almost everything the user experiences, what he/she sees, hears and feels, is real. There are new state-of-the-art interaction methods emerging that are based on an increasing amount of virtuality. The amount of virtuality in this context means "the amount of non-realness". In HCI's this is a measure for the amount of artificial, computer generated stimuli (sensory input), presented to users. This can be varied from one end, reality, where everything is real, to the other end, a virtual reality, where everything the user experiences, what he/she sees, feels and hears, is computer generated [17].

Augmented reality

Augmented reality is an interaction style with a limited amount of virtuality where most of the stimuli for users are real and only a few of the stimuli for users are generated by computers [15] [17].

Augmented reality can, for instance, enable users to see the real world with an overlay of additional information [15]. This is mostly done using heads-up displays or semi-transparent head mounted displays. For instance, it might allow a user to see wires and plumbing while looking at a wall through semi-transparent eyeglasses. It might let a surgeon look at a patient while at the same time looking at a transparent overlay of an x-ray to help locate a tumour.

New meeting technologies are proposed [17] that fall into this category. A few of those technologies might be used to assist users of meetings in real time by showing influence hierarchies or argument structures. In addition, meetings could be assisted afterwards by e.g. automated summaries of meetings to review later or for participants to be able to join a meeting later and catch up. Most of these technologies however, are far from mature and can't as yet be used in a real production environment [17]. In addition, some, even more futuristic visions are possible. These use electronic glasses, 3D displays or holograms to superimpose remote participants on the real environment as if they were there [17].

Virtual reality

Virtual reality is an interaction style where part of the real world (or an imaginary world) is represented as an immersive virtual 3D world. It has a high amount of virtuality where almost all of the stimuli for users are generated by computers [15] [17].

In virtual reality, the display is not a simple 2D plane, but a head mounted display or a room with one or more projectors, simulating a 3D environment in a more realistic way [15]. Mostly, some form of 3D actuator like data-gloves, movement detectors etc are being used to adjust the environment to the movement of the user. This is called an immersive environment [15]. This can allow people to be trained accurately for a number of difficult circumstances (e.g. a flight-simulator). This can also allow people to be virtually present in one room while actually be on different continents. Drawbacks are that it is still quite expensive, might require a large hardware setup [15].

The above mentioned drawbacks can be avoided by using a much less immersive 3D representation on 2D monitor, but that has the disadvantages of the above mentioned direct manipulation 3D interfaces: It might lead to slowdowns because of increasing disorientating navigation, too complex user actions and annoying occlusions[15].

3D virtual reality is mentioned in [17] as a possible way to do teleconferencing. Second life is such a 3D virtual world that can be used for that purpose. It is founded by Linden Lab of Linden Research and has a steeply rising user base that now reaches over 2 million. Second life uses a representation of the 3D virtual world on a 2D monitor as mentioned above. It is even already being used for educational purposes [24]. Therefore, Second life is further explored below.

The Second Life software provides free accounts with the ability to design, integrate and texture structures, furnishings, clothing, and avatars using in world resources. Sandboxes offer open spaces for creativity, collecting more free items and sharing ideas. The resources available to SL users include whiteboard and communication tools, language translators and information centres. A wide range of free software tools for Second Life is now available. Next to that, support from a variety of educational groups is available [24].
In Second Life, people use an avatar to represent themselves. They can define how their avatar looks like and they can create all kinds of virtual 3-dimensional objects using primitives like cubes or cylinders. The 3D environment is organized in the form separate simulated environments, called islands [24].

3D applications like second life can’t run on any old computer or operating system. They need a video card equipped with a 3D accelerator and a current driver capable of rendering 3D in hardware to be able to render images fast enough to be usable [33].

However, at close inspection, its educational uses as presented in [24], have more to do with learning how to create and operate objects in 3D than using it as a virtual teleconference or tele-classroom environment. In addition, it seems the user-base for Second Life as first reported by Linden Labs is not as large as first advertised [40].

Problems with complex user actions as mentioned above and in [15] might in the future be avoided by using cameras to perform tracking of hands and to estimate human poses, like proposed in [20]. This would render controlling an avatar in, e.g. Second Life a lot easier and a lot more natural. For the moment, these problems prevent it from being a very practical tele-classroom environment.

Conclusion

There is not always exactly one perfect choice for an interaction style. In many cases, a user interface will consist of a blend of a few of those styles, directed at different levels of functionality and user-experience. For the HCI of the old DTC system, the interaction style was mostly direct manipulation, combined with pull-down menus (see Section 4.5). Judging from the above, this is still the best choice for the HCI of a new DTC system.

9.3.2 Interaction devices

Interaction devices are devices to enable interaction between the user and a computer system. These include: keyboard and keypads, pointing devices, audio devices, displays, cameras and printers [15].

Keyboard

The keyboard is still one of the most used, although for small devices technologies like Palm's graffiti might be as good [15].

Pointing devices

Pointing devices can be divided in direct and indirect control devices. Direct control devices (e.g. light pen, touch screen) are very easy to learn, but may obscure the display while using them. Indirect control devices (e.g. mouse, trackball) take more time to learn, but do not have that disadvantage [15].

Audio

Audio interfaces to reproduce and record sound are these days a standard part of almost every workstation. Audio can be used to give feedback. This can be done by generating simple beeps, but nowadays this is done mostly by reproducing high quality familiar sounds during certain HCI-events. These can be viewed as auditory icons [15].

An audio interface is of course necessary in teleconference situations where an audio floor is present. For usage of sound during teleconference meetings, a number of tips are mentioned in [32].

- Avoid echoes. This can be done by using a headset (this usually gives the best results) or by using special echo-cancellation equipment when using normal microphones.

- Minimize background noise: It is strongly advisable to use a room that has only a minimal amount of background noise (by e.g. switching of fans and air-conditioning systems) and to prevent a room sound to "hollow" by e.g. finding a room with less straight corners, mounting objects to the wall, putting carpet and curtains in the room etc.

Speech recognition

Audio interfaces can be used for speech recognition. This concept has been around for ages, but its progress is slowly and there are still many technical difficulties preventing reliable performance, especially in more or less noisy environments.

The problem with speech recognition is not only technical. Voice commanding is more demanding on the users working memory that hand/eye coordination is [15].

In addition, speech requires the use of limited brain resources while hand/eye coordination is processed in another part of the brain. This can, for instance, in parallel do problem solving and planning while doing hand/eye coordination tasks. Therefore, using a speech-interface may be more disruptive to users while carrying out tasks [15].

Speech generation

Speech generation is also possible using audio interfaces. This is technically reliable, but also has practical problems [15]. Speech output has a much slower pace compared to visual output. Speech is ephemeral in nature (you can't 'glimpse back' to what just have been said). Also, speech output has difficulties providing an interface to scan/search large amounts of data.

Displays

Desktop displays

Desktop displays are widely used. Earlier there were only CRT's in the range, but nowadays they are more and more replaced by LCD's. A Common resolution for LCD's these days is 1024 x 1280 [15].

Some other technological alternatives are Plasma displays, LED-displays, electronic ink and Braille displays, but they are a lot less common and mostly still in development and or expensive. All these displays are relatively small [15].

Large displays

Large displays have been increasingly used. Especially since beamer technology has become affordable. They are not only used in command centres, but also during meetings and during education. When those displays are used interactively, normal indirect control devices like the mouse are impractical and are replaced by, for instance, a touch sensitive screen [15].

Also interactive elements like pull-down menu's need to be redesigned (e.g. by introducing a flow menu) since menus on a large touch screen must be compact [15].

A special example of such a large display with an integrated control device is the SMART board that provides a large touch-sensitive screen on which a computer image is projected. Fingers can be used as a pointing device, but it also provides digital coloured pens and a digital eraser to simulate a traditional whiteboard [15].

Small displays

Small displays are displays of a small size and with a low resolution like 240x320 pixels. These displays are used in mobile devices like PDA's and smartphones. HCI design for such displays needs to be custom designed to be optimized for such low resolutions [15].

`

Heads-up and head mounted displays

A heads-up display projects information on a partially silvered windscreen of an airplane or a car [15]. It is used for an augmented reality style HCI (see Section 9.3.1).

A head mounted display shows images on a visor or on glasses worn by the user. In most cases, it also contains tracking sensors to be able to adjust the images to head-movement [15]. When the display is (semi) transparent, it can be used for an augmented reality style HCI (see Section 9.3.1). When it is non-transparent it is mostly used for a virtual reality style HCI (see Section 9.3.1).

Printers

Printers are of no concern as yet for a DTC system.

Cameras

Cameras can be used for electronic recording of images and for transmission of images during a teleconference meeting [15]. Web-cams are mostly used these days for this purpose. Web-cams are cameras designed to be directly connected to a computer. Their resolution is mostly 640x480 pixels and they are now widely available on the consumer market [47].

For usage of light and sound during teleconference meetings, a number of tips are mentioned in [32] A few important ones are:

- Light: Most important as light is concerned is that the level of light is less important then the distribution of light. To light a face during teleconferencing one should never use back lighting or light from below, but light from about 60 degrees above.

- Viewpoint: The best view for a camera is when it is positioned somewhat above eye-height.

Image recognition

Cameras can also be used for image recognition. Image recognition is one of the input media that can be used in ubiquitous computing [15].

One of the applications of image recognition may be to assist people transmitting a better image. An application of rudimentary image recognition usage is shown in [19] with the usage of general face detection algorithms for camera direction. This could, for instance, be used in a future DTC system to assist lecturers by making sure their face is kept in the image when they move.

Other examples of the use of image recognition and detection are given in [20] in which a software utility to control an image-processing library, Parlevision, is described. This enables the tracking of hands, the estimation of human poses etc. In a future DTC system, something like this might be used to detect a student raising his hand to pose a question without the need to press a key or move a mouse.

Special purpose devices

There are some novel, mostly special purpose devices. Examples of these are eye tracking systems (mostly a research tool, not reliable to control a UI unless combined with manual input), multiple degrees of freedom devices (low precision, slow responses), data-gloves (heavy, confining) and (body) motion sensors [15].

Ubiquitous computing and Tangible user interfaces use e.g. video cameras (as already mentioned above) and motion sensors to track and sense user movements and let systems respond to them [15].

9.4 Developments in programming and software engineering environments

These days there are a large number of alternatives to develop an application. A number of these alternatives that can be used to implement a HCI for a multimedia teleconference application will be presented next. Some alternatives exclude each other, but many can be combined to make use of each of their strengths.

9.4.1 Scripting languages

TCL/TK

The HCI interface of the old DTC system uses TCL/TK. In the days it was designed, this was one of the more advanced methods for building HCI's. This was especially so because it has a high level in defining graphical elements of user interfaces [15]. The main drawback of not having a design-tool was overcome by using GuiBuilder (see Sections 2.4 and Appendices B and K).

TCL/TK still has its merits for fast prototyping, but it has not become a standard, especially the TCL-part [15]. It requires non-standard runtime libraries to be installed and its development and porting to different systems has been slowed down.

JavaScript, Perl/TK, Python/TK and VBS

TCL/TK has given way to other scripting languages like Perl/TK, Python/TK, VBS and JavaScript [15]. JavaScript is part of AJAX (see Section 9.4.2). These languages are all still in use and, except for VBS, can be used on different Operating systems. Most of these languages have no high level definition options for graphical elements. Also, GUI design tools for those languages are mostly not available.

9.4.2 AJAX

AJAX consists of, amongst others, JavaScript, but is more than that and forms a new way of combining a number of existing technologies. Therefore, it is mentioned in a separate section.

AJAX is an acronym that stands for Asynchronous JavaScript And XML. It is not a total new technology but an integrated combination of the following already existing technologies [22], [23]:

- Standards-based presentation using Extensible HyperText Markup Language (XHTML) and Cascading Style Sheets (CSS). XHTML is a markup language that is like HTML, but also conforms to XML syntax. CSS is a stylesheet language for (X)HTML.
- Dynamic display and interaction using the Document Object Model (DOM). This is a platform- and language-independent standard object model for representing HTML or XML and related formats.
- Data interchange and manipulation using Extensible Markup Language (XML) and Extensible Stylesheet Language Transformations (XSLT). XSLT is an XML-based language used for the transformation of XML documents into other XML or "human-readable" documents.
- Asynchronous data retrieval using XMLHttpRequest (XHR). This is an API that can be used by JavaScript, and other web browser scripting languages to transfer XML and other text data to and from a web server using HTTP.
- JavaScript is binding everything together.

An AJAX application eliminates the continuous need to reload whole web pages during interaction of classical web applications that slows user interaction down. This is done by introducing an intermediary, an AJAX engine, between the user and the server.

Instead of loading a webpage, at the start of the session, the browser loads an AJAX engine, written in JavaScript. This engine is responsible for both rendering the interface the user sees and communicating with the server on the user's behalf. The Ajax engine allows the user's interaction with the application to happen asynchronously, i.e. independent of communication with the server. See Figure 51 and Figure 52 for an illustration.

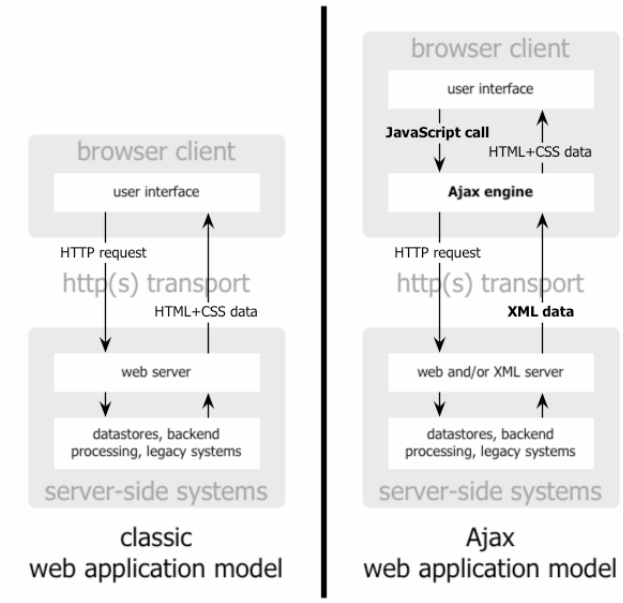


Figure 51 The traditional model for web applications (left) compared to the Ajax model (right).

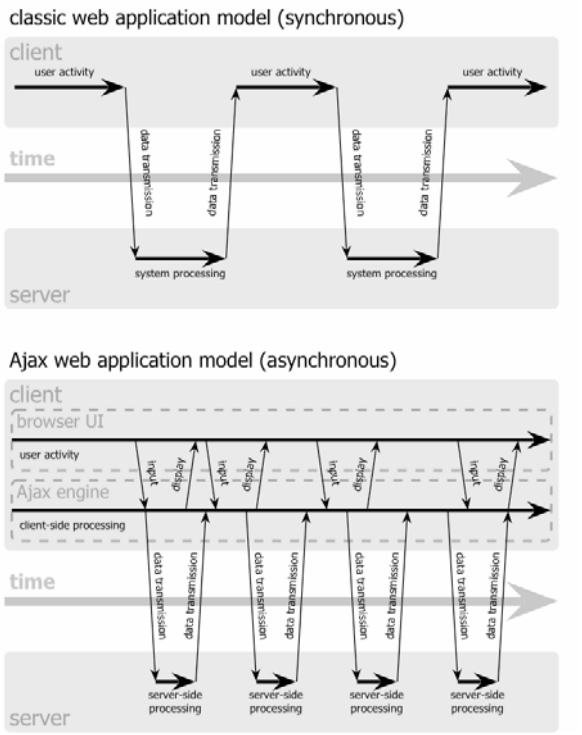


Figure 52 The synchronous interaction pattern of a traditional web application (top) compared with the asynchronous pattern of an Ajax application (bottom).

An ongoing development is the shift of using dedicated programs on workstations towards browser-based programs using AJAX. Google has already built an office-suite using this technology. Since a DTC-system requires enrolment in some kind of central system, this system might also supply the UI itself to the workstations.

It seems difficult at first to use AJAX technology for real-time multimedia applications, but it is possible. In [35] is described how AJAX can be combined with streaming video in the web browser Firefox. Furthermore in [36] Microsoft explains how to control an embedded Media player in their web browser Internet Explorer using JavaScript. Therefore, this must be possible.

Design tools and frameworks for AJAX are rapidly maturing and are widely available. Surveys and comparisons for AJAX frameworks are given in [27], [29], [31] and [34]. One of the common ways for AJAX is to communicate with a server running PHP and MySQL [38].

In [31], a classification for a number of frameworks has been made between server centric and client centric frameworks. The main difference is that server-centric frameworks are very complete and integrated, but are difficult to combine with other applications. This is a problem, since it is vital for a DTC application to integrate an AJAX application with media-players on the client side. Besides, it is also advisable to integrate the server side into an enveloping framework like Moodle (see Section 9.4.6). Another advantage of a client-centric development environment is that it can be used to design an integral and functional part of an application on the client side without the need for a complex backend server to be configured.

The Google Web Toolkit (GWT) [29], [30], [34] is a very promising framework that can be combined with a limited set of functions from Eclipse [21]. Unfortunately, according to the classification in [31], it is more server-centric oriented and as such it might be less suitable.

From the platforms that are client-centric according to [31] and are freely available, Tibco General [63] seems the most promising. It combines a graphical design tool for user interfaces, the freedom of open source with the quality and refinement of a commercial product [27].

9.4.3 Java

Java is a machine independent language developed by Sun [15]. Integrated development tools and libraries supporting Java like, for instance Jbuilder [15] or Eclipse have matured. Especially Eclipse is very versatile and contains a lot of libraries and modules, like, for instance, SWT. SWT assists in making applications that have a user interface that blends in as much as possible with the operating system and the graphical environment the application is run on [21], [30].

9.4.4 Flash

For rapid prototyping, it is also possible to use interactive animation design-tools like Macromedia director and Flash [15]. Flash is now part of Adobe, but has become very powerful and is good at combining web-based interaction with multimedia [37]. This is also supported by the fact that 2 of the applications mentioned in Section 9.5 are based on flash. Unfortunately, Flash authoring tools are commercial and must therefore be purchased. Our standard systems at the university don’t have those tools available.

9.4.5 Dot Net

Dot Net is build by Microsoft and is not a programming language or a complete development environment, but a set of libraries usable to create HCI’s [15]. It can be combined with different programming languages like c, c#, visual basic etc., but it is only available for MS-windows operating systems. The open source and cross-platform alternative, Mono, has come a long way, but still is not complete [39]. Dot Net is already being used for teleconference implementations (See Section 9.5.6)

9.4.6 CMS

At this moment, the internet, and especially World Wide Web, is being used for many purposes. Many web-based frameworks have been created to accommodate those different purposes. One of these purposes for which multiple frameworks have been created is content management. A CMS is a Content Management System. It is a system to create, edit and display digital content during its lifecycle. In many cases, a CMS enforces some kind of style using predefined templates. It can offer editorial functions, but that is not a requirement [28]. A web based CMS is used to manage content in sets of web pages. It requires only a browser to access its functionality and does not require users to have direct knowledge of HTML. Joomla is an example of such a web-based CMS [51].

Some variants of CMS are designed to be even more specific and to provide a platform for e-learning. Both in [25] and in [26], Moodle is mentioned as one of the best and most flexible platforms for e-learning. This is extra supported by the fact that one of the inspected current teleconference applications, Dimdim (see 9.5.5), offers integration with Moodle.

In the old version of the DTC, provisions were already made to integrate other modules into it and also to integrate it eventually into a kind of course registration system. It seems like a logical continuation to strife for an eventual integration of a new DTC system into one of these frameworks.

9.4.7 Conclusion

Using TCL/TK for a desktop conference system isn't the best choice anymore. Using Flash seems to be a very good choice. Unfortunately, this is not available as engineering environment at the university at this moment. AJAX in combination with a separate media player and usage of a free design tool like Tibco General also holds many promises. Integration with an educational CMS framework like Moodle seems to hold even more promises.

9.5 Developments in the area of teleconference systems

A number of current teleconference applications were reviewed in order to get an idea what is possible at the moment and how it was done. Most were selected by references from examined documents, standard availability at the University of Twente, or oral recommendations. A few were found using the criterion of modifiability. For more details, see Appendix M.

9.5.1 Marratech

The Marratech 6.0 (and 6.1) teleconference system exists of client and server software that is made available for a number of platforms (e.g. Windows, Linux, Mac OS X) mostly because much of the software is written in Java. On the server side, it is compatible with e.g. SIP and H323, making it possible to invite users of other (hardware based) teleconference systems to a teleconference. SIP, the Session Initiation Protocol is used to authenticate and negotiate capabilities between peers at the endpoints networks [58]. H323 is an ITU protocol that originally was designed to provide an IP transport mechanism for video-conferencing. It has become a standard in IP-based video-conferencing equipment [58].

The client software of Marratech is standard available on workstations of the University of Twente, that has an account with Marratech to have meetings hosted.

The client enables users to send and receive video, audio, and text. In addition, it has one shared whiteboard available. Furthermore, it enables application sharing by using VNC [60], making it possible for users to view a live application on one system and even handing control of that application over to another user. The standard user interface consists of one window, which is divided in different areas

Teleconferences are organised in virtual meeting rooms. In each meeting room, users can have one of 4 different roles: Moderator, Presenter, Attendee and Listener.

It has a rudimentary floor control mechanism by letting the moderator change roles for users, but a floor like audio is uncontrolled. It has no floor request mechanism.

9.5.2 Adobe Acrobat Connect

This commercially available teleconference system uses a current web browser in combination with the Adobe Flash player as its software platform, making it capable (especially for the client) on many systems to use audio, video and text communication. Windows and Apple OS X systems have some extra possibilities like file-uploading and screen-sharing. A predecessor of this system, Adobe Breeze, is recommended and used by ‘Expert on a distance’ [32].

Acrobat Connect organises meetings into virtual meeting rooms. Each user in such a room can have one of tree roles:

- Host: Can set up a meeting and invite presenters and participants, can add content to the shared library, can broadcast audio, video, can use text chat and can share content. He/she can promote participants to presenter or can give enhance permissions to participants. A host can also organize participant-polls.
- Presenter: Can broadcast audio, video, can use text chat and can share content.
- Participant: Can use text chat and can view and listen to broadcasted audio, video and content.

The role of each user is visible in the user-list and users do have a status-icon they can set indicating for instance they have a question. The host can rest that status at will.

Acrobat Connect has some means for floor control: As indicated above, the host can promote participants to presenter and vice-versa. In addition, the host can give participants enhanced

permissions for certain media or can take these away. This can e.g. be done in response to a question-status set by a user, although status showing and resetting is done independent of assigning roles or permissions.

9.5.3 Tandberg hardware-based teleconference system

This is an independent hardware based system, providing users with a remote controllable pan-tilt-zoom camera and a build-in room microphone. It is used, amongst others, in the project 'Expert on a distance' [32]. Up to 4 systems can be connected using the H323 protocol without the use of a server to provide a teleconference system. It provides users with a one large view of a video transmission, accompanied with the audio combined with this transmission. A small view of each of the video streams of the participants, including their own transmitted image is also available. By choice, audio and video can be transmitted from an auxiliary input (e.g. a laptop or a DVD-player), instead of using the build-in camera.

During a teleconference, no user has any explicit assigned roles.

This system is more aimed at lecturer-to-group or group-to-group situations. Therefore it has no floor control mechanisms. It can not always be used stand-alone. For everything but direct audio and video, a separate PC is still needed. Pan-tilt options are mostly not a requirement for single desktop users.

9.5.4 Microsoft Live Meeting

This is a hosted application. It is more an extended collaboration tool, making it possible to show presentations, to use text chat and to organize a meeting. Voice connections are done using external VOIP connections from e.g. British Telecom (BT), but can be controlled from within live meeting. It was reviewed because Microsoft in the past bought the CuSeeme software, upon which the network layer of the old DTC system was partially based.

9.5.5 Dimdim Webmeeting

This is an open source webmeeting teleconference application that still is in an alpha phase. Since it is open source, it can be freely modified if one has the proper authoring tools. It is using the Flash plugin from Adobe on the client side and it uses Java on the server side. It has options for communicating using audio, video and text chatting. It has no shared whiteboard (yet), but can show documents like PowerPoint or Word, that have been uploaded (in advance or during the meeting). Furthermore, it has the possibility to share the desktop. It lacks many features for dynamic floor control for all media and has no floor request mechanism.

Dimdim has for now only two roles for users: Presenter and attendee.

An interesting feature is that Dimdim provides an integration-pack (still very experimental) for Moodle (an open source web-based e-learning platform).

9.5.6 Vmukti

This is open source webmeeting teleconference application that was formerly known as 1Videoconference. Since it is open source, it can be freely modified, if one has the proper authoring tools. It is build using .NET 3.0 and is Windows 2000/XP only. For VOIP/video connection, it relies on a standard SIP service like Asterisk that needs to be installed separately.

Vmukti has options for communicating using audio, video and text chatting, shared whiteboard, shared applications, and shared desktop. It lacks many features for dynamic floor control and has no floor request mechanism.

A very interesting feature, like Dimdim, is that also Vmukti provides an integration option for Moodle (an open source web-based e-learning platform).

9.5.7 LearnLinc

LearnLinc is part of a suite of teleconference applications from iLinc, but in this report only LearnLinc will be reviewed. A previous version, Dijdidact, that was available in Dutch, was recommended during talks with a friend, Ellen Rusman, who works at the Open University in Heerlen.

LearnLinc works only on PC's running Windows 2000 or XP. It uses a browser (IE or Mozilla) to login to a teleconference session using a login webpage, the communications centre. This can contain several meetings and schedules for meetings. It also contains functionality to define meetings, meeting schedules, predefined content, participants, communications settings and security settings.

LearnLinc is organized as a toolbar on the left of the screen, with regions that can be collapsed to a small bar or expanded to the middle of the screen (see picture). Standard options include document sharing (e.g. PowerPoint), desktop sharing, browser sharing, a shared whiteboard, feedback questions, multiple choice questions lists and surveys.

Standard there is only an option for audio using telephone conferencing available. Audio- and videoconferencing using the internet is optional and requires additional modules.

In LearnLinc, there are three roles for users:
- Leader: controls the whole session.
- Assistant: can do everything the leader can do except the following: Grade participants, create breakout groups and speak when he or she does not has the floor.
- Participant: participates in the session and can ask floor control to the leader.

There are means for floor control present in LearnLinc. The leader takes the floor automatically when he or she joins the session. The leader can pass the floor to any participant and take the floor back at any time. While the participant has the floor, he or she can talk and launch content and applications. The leader's microphone is always on, and he or she can talk and launch content even when someone else has the floor.

Participants can request the floor by "raising their hand" (pushing the "raise hand" button). The leader sees a raised hand icon next to the name of the participant in the participants-list and sees raised-hand counter increased by one. He or she can acknowledge that by giving the floor to that participant or acknowledge this verbally. There is an option to force the lowering of all raised hands. There is no way of reviewing the order the requests arrive in.

Text chat is not part of that floorcontrol mechanism. A participant in a LearnLinc class can always send messages to all participants at once or to one participant in particular. There is however a separate option for the leader to limit text chat to the ability to send messages to the leader, assistants, and the current floor holder.

In addition to the central meeting (class in LearnLinc) there is an option to organize breakout groups. This is a meeting room to which the session leader sends a group of participants to work collaboratively. This continues until they decide to return, or until a set period of time has expired. In the breakout group, anything is possible that can be done in the main session, including sharing content, applications etc. When the breakout group time limit elapses, all participants in the group automatically return to the main session.

In LearnLinc there are options for the leader and assistants to track the participants engagement (a kind of activity-meter that indicates a participant is active during the last minutes on his / her system or not). He or she is also able to look at the desktop of a user without permission (a function called glimpse).

A leader can assign, edit and view grades for all participants during or after a class session. Participants can view those grades from within the communications centre.

9.6 Results

A lot has changed. It seems that choosing Tcl/TK for HCI implementation of the old DTC application was a good choice for a long time. Now, new tools and environments are emerging.

A new design of a DTC should use a number of nowadays available principles and guidelines to arrive at a better and more consistent user interface.

As a suitable design platform, Flash seems to be particularly promising. This is however not available at the university today as design platform at this moment. AJAX in combination with a separate media player and usage of a free GUI design platform like Tibco General also holds much promise. In addition, integration with an educational CMS framework like Moodle seems to hold even more promises.

Alternatively, it is possible to take one of the existing teleconference applications and use that as a starting point for a new DTC system. As far as functionality is concerned, LearnLinc has most in common with the DTC system and even has means for floor control present for some floors (see also Section 11.7).

In the future, a number of new interaction types and facilitating services might become available. It is possible to anticipate for them with recommendations and to set up some experiments in a university environment, but most are not yet ready for real usage.

10 New constraints and user requirements

10.1 Approach

In this chapter, the old constraints from Chapter 2 and the user requirements from Chapter 3 will be newly re-evaluated. The constraints set boundaries for design and implementation and they define the demanded functionality. The user requirements are an elaboration of this functionality. They define the functions the HCI has to offer in more detail and in relation to each other.

This re-evaluation will be done in perspective of the scope of this part of the report, changed conditions and new knowledge, both gained in chapter 9. It will result in new, adapted or recommended constraints and requirements.

10.1.1 Old constraints

The old constraints are presented in Section 2.5. They are split into hardware, software, and functional constraints. Those included demands that the software had to run at least on Sun Solaris workstations, made use of an existing implementation of a session layer, had at least to provide a teacher with the means to transmit audio and visual information etc.

10.1.2 Old user requirements

This is a summary of the user requirements for the DTC system as given in Chapter 3.

These requirements are given in the form of descriptions of scenario's using floors, actors, roles and tasks (See Section 3.3.2 and Section 3.3.6 for definitions).

The DTC system has only one scenario as given in Section 3.3.3. This is the lecture scenario. This scenario has a number of actors that can have 2 different roles: Teacher or Student. The role descriptions for this scenario are: "There is one teacher who gives a lecture" and "The Students attend this lecture". This is further elaborated in Section 3.4.4 to Section 3.3.7.

To make a DTC system more universal, its requirements are also based on a more generic Desktop Teleconference (DT) system as given in Section 3.4.

To base the DTC system on this generic DT system, a mapping has to be made from the roles tasks and scenarios of this DT system to the actual DTC system to be designed. For more details, see Section 3.5.

To describe the actors, their roles, their mutual interaction and their interaction with the system itself more clearly, point wise scenario-definitions, task analyses and Entity-Relationship (E-R) diagrams were used throughout Chapter 3. Detailed task analyses can be found in Appendices C, D.

10.2 Adaptation of constraints

In this section, the old constraints as given in Section 2.5 are discussed and, where needed, adapted to the current situation. Like in Section 2.5, these are split into hardware, software, and functional constraints.

10.2.1 Hardware constraints

Workstations

SUN workstations are hardly used anymore at the University of Twente, so it should not be a requirement anymore to use SUN's to develop (part of a) a DTC application. The most supported environment within EWI at this moment is an Intel x86 compatible PC running MS windows XP, so every newly built prototype should be able to run at least on such a system. This is also the system that is used by the majority of computer-users [59]. Even then it is advisable to make future designs as platform independent as can be. The Operating systems can still vary (Windows XP, Vista, Linux) and not all hardware is always compatible (Apple, mobile devices, etc.). This might not be possible for the DTC system as a whole, since it is still dependent on hardware for e.g. video and audio acquirement, but the Graphical User Interface (GUI) part of the HCI for the DTC system can be made machine independent.

Special hardware

Cameras do not need special video-digitising hardware anymore, as SUN provided for its workstations. Cameras are now available as USB webcams at very low cost that can be connected to any current PC. In addition, soundcards, with in- and output are now a basic part of every PC system. It could therefore now be possible to require each user to have video and audio available.

The old session layer implementation used by the DTC system was designed to make use of special ATM network hardware to reach the transport performance needed (see Section 1.3.5, Section 5.5.2 and Appendix J). As stated in Section 8.3.2, normal networks today are fast enough for video transmission.

10.2.2 Software constraints

The session layer implementation

A constraint for the design of the DTC system in Section 2.5.2 was to make use of a separate session layer software implementation [1] [2] [13]. This implementation was based on old operating systems (SUN's Solaris 5 and later also on Windows NT 3.51 and 4.0, see Appendix I). These are all not available anymore. It is still possible to run the modified CU-Seeme reflector on a current Solaris system, but the client side of the session layer implementation refuses to run on current Solaris and Windows systems. This means it has to be abandoned or it must be updated.

Updating is almost impossible: Not all parts of the source codes of that session layer implementation are available anymore, so it is not possible to continue with this implementation by adapting it (see Section 8.4.1). In addition to that, the session layer implementation uses Vendor-specific code (SUN's CellB) to encode video streams that need special hardware that has already been discontinued for years.

In view of the above, this session layer implementation will have to be abandoned. A number of standards for multimedia transport, coding and decoding have been emerged [18] [50] [49] [38]. These can be used to transport multimedia and can be used to build a new session layer implementation.

TCL/TK

As already stated in Chapter 8 and Chapter 9, the choice for TCL/TK as prototype and development environment Guibuilder is at this moment a less obvious one.

As indicated above in Section 10.2.1, it is still advisable to make future designs of a DTC system as platform independent as can be. Furthermore, its design should be possible with means directly available at this university. In addition, it should be possible to integrate multimedia control directly with the rest of the GUI. From Section 9.4 and Section 9.6 we can conclude that, given these constrains AJAX is one of the best candidates to be used for a future DTC system and it is recommended to take usage of AJAX as a software constraint.

Logistic support

The logistic support system presented in [5] has not become a standard and it is not necessary anymore to integrate user data from this system in a DTC application. Instead, other course and content management systems have been created that support this kind of functionality and much more. In Section 9.4.6, Moodle is given as the best alternative and it is recommended to set integration with Moodle as a software constraint.

10.2.3 Functional constraints

The functional constraints presented in Section 2.5.3 describe the demanded functionality to be provided by the HCI of a DTC system. This is reflected in the user requirements in Chapter 3 that describe the functionality of the HCI of the DTC in detail.

Within the scope of this report, it is not possible to do extra research to refine these constraints, but in Chapter 4, suggestions were already made for a next design cycle. These can be used. Furthermore, some of the most obvious changes in hardware over the years as given in Chapter 8 and Chapter 9 can be used to refine these constraints.

Media facilitation

The old functional constraints demanded at least facilitation of audio and text transmission for teachers and of text transmission for students.

One of the most obvious changes in hardware in the past 10 year is the improved options for capture and transmission of video images. This is widely available these days (see Section 8.3.2, Section 8.3.3, and Section 9.3.2). In addition, audio interfaces are today a standard part of most workstations. Furthermore, when looking at current teleconference systems, the majority of these use a central window to be used for graphical presentations (e.g. Power Point). In many cases, it can also be used as a shared whiteboard.

It is therefore recommended to sharpen the functional constraints to demand at least audio, video, presentation and text transmission for the teacher, and demand text and audio transmission for students.

Floor control

The only part that is outdated in the part about floor control is the remark that floor control was already partly implemented in a session layer implementation because this (see above in Section 10.2.2) is not functional anymore. This session layer most be designed and build anew, along with the HCI. Next to transfer of multimedia, it must support floor control.

As indicated in Section 4.2, the functionality of the DTC system is smaller than that of the generic Desktop Teleconference (DT) model introduced in Chapter 3. In Section 4.6, suggestions were already made for a next design cycle that can be used. Most of these concern aiming the design towards such a more generic (DT) system. For a total new design, it would therefore be recommended to extend the functional constraints to demand the more universal functionality of a generic teleconference system by extending floor control options.

A recommendation, derived from Section 4.6 is to extend the functional constraints so that it should at least be possible to assign floor control for different floors to different participants or to free a floor of all control.

Another recommendation that can be derived from Section 4.6 is to extend the requirement for floor control from only write access towards read access for all floor participants.

Using ITU document T130

The used ITU document (T 130) [12] defines a standard architecture and conference model. The old user requirements were for an important part based on this document and it is beyond the scope of this part of the report to consider changing this.

10.3 Adaptation of user requirements

In this section, the old user requirements as given in Chapter 3 and Section 4.2 are discussed and adapted where needed. This is done using the adapted functional constraints as given above in Section 10.2.3 in combination with suggestions from Section 4.6. In addition, when already discovered, extra suggestions for user requirements in Chapter 9 are included. More improvements might be found using extended research, but that is beyond the scope of this part of the report.

10.3.1 Keeping the old requirements

As the functional constraints are kept, and are only extended for some details, most of the old requirements as given in Chapter 3 and Section 4.2 can also be kept as they were, since they reflect those constraints. With respect to the changed functional constraints are they to be changed. This means the removal of some previously restrictions as given below.

10.3.2 Removing restrictions

As already indicated above in Section 10.2.3, the functionality of the DTC system is smaller than that of the generic DT model. This is because there are only two actors defined. Furthermore, to fulfil the old functional constraints, not even all functionality of the DTC system had to be implemented. Therefore, requirements for a limited implementation of a system according to the generic DT model were suggested in Section 4.2.2 by setting restrictions on the requirements for the design of the old DTC system. These were:

- Since the teacher is both groupmeeting controller and floor controller for all floors, it is not necessary for the HCI to offer functions to assign floor control to groupmeeting participants. The teacher is only concerned with floor control. Therefore, the decision was made to implement group control functions only as a mock-up.

- Because each student has to be able to follow a lecture using all available floors, it is not necessary to provide functions to request read access for floors. Instead, sink-, or read access- rights should be given to a student by default for all floors. The decision was therefore made to let all users always have sink-rights.

As mentioned in the functional requirements in Section 10.2.3, suggestions in Section 4.6, were already made for a next design cycle. These concern, amongst others, the removal of the above-mentioned restrictions for a DT system. For a new design cycle it is recommended to do so.

10.3.3 Extra user requirements

In Section 4.6 a detailed description is given to extend the requirements with more local media control. This is needed for local source selection for someone who has the floor. It is also necessary to control remote sources when a floor user can join more floors of one type. This is given in terms of the already partly designed HCI and not in a HCI independent task analysis. It is recommended to do this first.

In Section 4.5.3 the decision was made to show floor requests not only in the floor request-list for the floor controller, but also as request indication in the floor participants list. It increases the awareness of the floor controller (See Section 9.2.6) since it combines an overview of all floor participants with an indication of which of those users have requests or have floors granted. This is implemented as such in the old DTC system. It was also found to be implemented in this way as (the only) means of showing requests in both Adobe Acrobat Connect (see Section 9.5.2) and LearnLinc (see Section 9.5.7). It is therefore recommended to include the above extra request indication in the user requirements.

During the study of current teleconference systems in Section 9.5, a feature of one of those examined systems seemed particularly useful. This is: showing requests from other floor participants for floor access like in LearnLinc (see Section 9.5.7). It is useful because it can be an advantage for a participant to see all pending requests. It increases the awareness of the users of the DTC system (See Section 9.2.6). Seeing other requests can, for instance, aid a user in the decision whether or not to initiate a new request. Therefore, this is an extra recommended requirement

11 Recommendations for a new HCI design of a DTC system

11.1 Approach

In this chapter, a number of considerations and recommendations for a new design cycle for the DTC system will be given. This will be done using the knowledge gained in Chapter 9 and the updated constraints and requirements as given in Chapter 10. Within the scope of the second part of this report, it is however not possible to fully design and implement an interface satisfying all those constraints and requirements. Therefore, the emphasis in this chapter will be placed upon general recommendations for a new implementation of a DTC system using current knowledge and technologies.

As has been indicated in Section 8.4.1, a combination of two strategies will be used in this report. Therefore, next to the above general recommendations, an existing teleconference application will be selected that best matches the requirements for a DTC system as given in Chapter 10. Subsequently, a number of recommendations will be given to improve this application in order to bring it even more in accordance with the requirements of a DTC system.

11.2 Interaction styles and devices

During the recent years, the number of interaction styles to realise a HCI has increased (See Section 9.3). Especially the increase in hardware possibilities and performance opens new paths to realise a HCI for a DTC system. A number of recommendations concerning those possibilities will be given in this section.

11.2.1 Use medium sized, individual displays

The old DTC system was based on the assumption that each user was seated in from of his or her own computer system. Each computer system was equipped with speakers a microphone and (optional) a camera. Users only interacted with each other using the DTC system running on those computer systems.

In Section 9.3.2 and Section 9.5.3, especially in ‘Expert on a distance’ [32] it seems that combining a more traditional classroom setup with the usage of large projection displays, (that can be touch-sensitive in some cases) might be very useful in a number of cases.

The most important argument against using this kind of setup in a HCI design for a new DTC system is that the requirements and design of the DTC system in the first part of this report and in Chapter 10 are based on using individual computer systems for each user. Therefore, it is recommended to keep a future design of a DTC system also based on individual computer systems with normal, medium sized displays.

11.2.2 Use a 2D direct manipulation interaction style

In Section 9.3.2, the option was explored to use a 3D virtual reality interface for a future DTC-system that might give a more natural idea of a virtual classroom. It seems possible to use an extendible application like 2nd life [24] as a base and go from there. However, in the same section, it was also doubted that, with the current state of technology of interaction devices, this will be successful. Especially 3D interaction using 2D computer displays is too cumbersome for many users to use with present interaction technology. 2nd life seems to be only useful for actual 3D content creation / operation.

It is therefore recommended to stay with a standard 2D direct manipulation based interaction style for a HCI design of a DTC system.

It is however possible to extend the requirements of the DTC system to enable it to be used in a situation that involves 3D content creation and/or operation, if needed. In that case, it is recommended to integrate part of the interface of the Second Life client into the DTC as a new type of floor, a 3D interaction floor. The imaging and operations of the client are then limited to a window within the 2D DTC system that shows the 3D virtual world of Second Life. Since floor control needs to be functional also for this type of floor, the integrated Second Life client needs to be modified. It must be able to block certain user-actions that manipulate content in the 3D world when a user has no source rights for this 3D floor. When a floor user is denied sink rights, the client must block the 3D view completely for this user.

11.2.3 Use traditional interaction devices in a traditional way

New interaction devices and new ways to use existing interaction devices, discussed in Section 9.3.2, like camera control by image recognition, and recognition of gestures, are currently in an experimental phase. These are therefore not recommended for today's DTC implementation. In the near future, however, it is recommended to use these as they come available for non-experimental use.

11.3 Modelling user interaction for the DTC system

Section 9.2.3, exposes several modelling and specification tools. These are methods like grammars, menu-selection trees and dialog-box trees, transition diagrams and state charts.

From Section 11.2 can be concluded that 2D direct manipulation is the recommended interaction style for a new HCI design for a DTC system.

In Section 9.2.3, it was already concluded that for this kind of interaction, using state charts is the best way to specify human computer interaction. Therefore, statecharts are recommended to be used for modelling user interaction during the design of a HCI for a new DTC system.

11.4 Visual appearance and operation

In this section, a number of recommendations will be given for the design of the visual appearance of the HCI of a new DTC system. References to the design of the old DTC system will be made where possible.

11.4.1 Use only one window, divided in sections

One important fact is that users can only remember 7 plus or minus 2 items [15]. While this seems adequate, the number of windows opened during a session with the old DTC system already reached this number. In the old DTC application, it was possible to open many windows like the floor-window (that itself is divided into several areas), video-windows, text windows, a whiteboard, a media-control window, and a few pop-up windows (See Section 5.5). Some of those windows might be on top of each other and thus might be forgotten. A user not always has to remember all those windows and its functions at once. Combine this however with the fact that each window contains a lot of constantly updated information and it might become too complex too remember and to manage. Especially if they all can to be minimized, maximized and positioned separately between windows of other applications.

Most modern GUI's of applications these days, work with a single container window that assists navigation and management of different parts. This is done by providing grouping by tabs, resizable sections within that singe container, and (fixed or floating) toolbars. This can be seen for instance by looking at the layout of the Marratech teleconference application in appendix B)

In the old DTC system, windows could be resized and moved around, but the layout within each window of the old DTC system is fixed, while users prefer a certain amount of customization. The above suggested container window combines the flexibility desired by users with a clearer way of organising different parts of the GUI.

It is therefore recommended to use one single container window divided in several customizable sections to organise the layout of the HCI of a future DTC system.

11.4.2 Use colours sparingly

The old DTC system was quite colourful. This was partly inspired by many motif-like applications with big coloured window-borders that surrounded it at the time. It might also be seen as a means to make grouping of some areas more clear, to make buttons stand out and to be able to distinguish windows belonging to the DTC application from others. Current guidelines give advice to use colour only sparingly. The advice is to mostly use it to draw attention (see [15]).

The recommendation for a new HCI design of a DTC system therefore is to first design the layout without any colour and to add this later, only where needed.

11.4.3 Use icons where possible

Icons are very versatile to represent all kinds of functions. In many cases they are much more compact that one or more words on buttons describing the same functions. This means they can leave more room for other parts of the GUI (See Section 9.3.1 and [15]). Icons also stand out more from the environment [15]. In the old DTC system, icons were only used as representation of the floor-access permissions for users. It is best to use them for more that only this single function. Icons are more compact and can, for instance, also be used within lists.

For a new design of a HCI for a DTC system, it is recommended to use icons where possible. A number of guidelines from [15] for the designing of icons are mentioned in 9.2.5. It is recommended to follow these guidelines.

11.4.4 Use pull-down menus only when needed

When designing a future DTC application, it is better not to present most of the functionality in a main pull down menu like the old DTC application had. Instead, it is recommended enable the selection of much used functions by icons (see Section 11.4.3). These can be combined with pop-up menus when needed. This increases the direct manipulation style of the HCI (See section 9.3.1) giving users more the idea they directly manipulate the properties of a system.

Attention should also be given to consistency [15] of menus when they are still used. For example: The old DTC application had the possibility to select actions on different users from within a popup menu in the user lists but also using the pull down menu. Unfortunately, to realise this technically in TCL, the selection sequence used in them main window of [person -> chose action on person in a pop up menu] was the opposite of the pull down menu.

For the design of a new DTC HCI, it is recommended to avoid the use of pull-down menus where possible and use icons instead. Where pull-down menus are to be used, care should be taken to preserve consistency.

11.4.5 Ensure keyboard operation for all major functions

When designing a direct manipulation HCI that also contains menus, it is part of guidelines given in Section 9.2.5 to give all frequently-used menu items their own direct keyboard shortcut. Especially experienced users use the keyboard more often while inexperienced users make more use of graphical interfaces (see Section 9.2.4). It could also assist users who can't use a mouse. Therefore, even without using menus it seems logical to give frequently used functions a keyboard shortcut. Keyboard shortcuts could, for instance be defined by giving each function a CTRL-key + letter combination.

In the old DTC system, not all important functions could be accessed in this way. Instead, the menu on the top of the window can be accessed by keyboard traversal (using the cursor keys) and by using the ALT-key and letter-combinations to navigate through menus (See Section 5.5.3). For a number of popup menus in the user-list and the request list this keyboard traversal was impossible.

It is recommended to give frequently used functions a keyboard shortcut in a new DTC HCI design. Furthermore, to make the HCI more consistent as is a design principle given in Section

9.2.4, it is recommended to make keyboard traversal possible in all menus (if any) for the HCI of a new DTC system.

11.5 Design platform

To implement a HCI for a new DTC system, a programming and design platform must be chosen. From Section 10.1.1 it is clear that using AJAX is a software requirement. For designing a HCI for a DTC system, in Section 9.4.2 it was already concluded that, form the available GUI design platforms, Tibco General was the most suitable candidate to be used. It is therefore recommended to use Tibco General as a design platform for a HCI of future DTC system.

11.6 An integrated web based system

As can be seen in Section 10.2.2, an additional requirement is that the DTC system can be embedded in Moodle, an educational administrative system. This provides means for teachers to distribute documents and for students to submit finished assignments. From Section 9.4.6, it is clear Moodle supports integration in many ways. From Section 9.4.2, it is clear that integration can be achieved by combining Moodle with applications mad with an AJAX design platform like Tibco General. It is therefore recommended to integrate a future DTC system with Moodle.

11.7 Recommendations to improve a current teleconference system

11.7.1 Selecting the most suitable application

Only one current teleconference system will be selected to give recommendations to improve such a system with functionality for a DTC system (See Section 8.4.1). To do this effectively, first the application has to be found that already approaches the DTC user requirements from Section 10.3 as best as possible.

Important aspects to look for are:

- Teleconference functionality (audio, video, whiteboard etc)
- Floor control functionality
- Floor request functionality
- Desktop usability

The current applications reviewed in Section 9.5 are the ones where knowledge with respect to the above aspects is available. One of these will therefore be chosen. The reviewed applications are:

- Marratech (Section 9.5.1)
- Adobe Acrobat Connect (Section 9.5.2)
- Tandberg hardware-based teleconference system (Section 9.5.3)
- Microsoft Live Meeting (Section 9.5.4)
- Dimdim Webmeeting (Section 9.5.5)
- Vmukti (Section 9.5.6)
- LearnLinc (Section 9.5.7)

Of these applications, the Tandberg system is not meant for desktop conference. Since the focus of this report is only on desktop conference systems (See Section **Error! Reference source not found.**), this system does not qualify.

Microsoft Live meeting is more meant for collaboration and lacks direct voice and video transmission functionality. Dimdim is still in an alpha phase of development and lacks some extended teleconference functionality like a shared whiteboard.

Marratech, and Vmukti have much of the desired teleconference functionality, but they lack some floor control mechanisms. Both have no floor request mechanisms available.

Adobe Acrobat Connect and LearnLinc both provide the desired teleconference functionality and good floorcontrol. Adobe Acrobat Connect has a partially implemented floor request mechanism present. LearnLinc has the most extended means for floor control and good floor request functionality available.

See Table 4 for a detailed evaluation.

Application	**Teleconference Functionality**	**Floor control functionality**	**Floor request functionality**	**Desktop usability**
Marratech	+	+/-	-	+
Adobe Acrobat Connect	+	+	+/-	+
Tandberg system	+/-	-	-	-
Microsoft Live Meeting	-	+/-	-	+
Dimdim Webmeeting	+/-	+/-	-	+
Vmukti	+	+/-	-	+
LearnLinc	+	+	+	+

Table 4: Evaluating current teleconference applications.

As can be seen from Table 4, LearnLinc is the most suitable application of the ones reviewed. For this application, a number of recommendations will be given in the next section.

11.7.2 Recommendations for the improvement of LearnLinc

Functionality

LearnLinc already has much of the extended functionality needed for a DTC system, yet some things are missing.

One of the things it lacks, when comparing the description in Section 9.5.6 and the user requirements in Section 10.3, is a proper floor request list with request summaries. Floor request indications are shown in the user list, but it lacks a separate request list. This could for instance be sorted in chronological order. It helps the teacher to make sure students requests are treated fair and in order.

To have the above mentioned request summaries available, an optional input field and /or popup window is needed to state that summary when requesting a floor.

According to the user requirements in Section 10.3, there should be functionality to refuse a floor request. This is in LearnLinc only available as a crude option to force lowering of all hands at once. It is suggested to make an option available for both the attendee-list and the request list to force lowering of only one specific hand. This could be accompanied with a message-option to give a reason. This was as such implemented in the old DTC system, but this is not a defined part of the requirements.

As can be seen in the user-requirements in Section 10.3, it is advisable to extend floor control possibilities to all floors available during the teleconference. This means floor control should also be possible for text-chatting. It has the extra advantage of making the GUI more consistent [15] by treating all floors equally. When the text chat floor is controlled, it is best not to allow any private chats to keep usage of text chatting consistent with floor control rules.

Extending floor control means also that an option for floor control should be available during breakout sessions. This might, for instance, create the possibility to create assisted breakout sessions in which each group is lead by an assistant. It is necessary to have a role available for leading a breakout session. It might be possible to use the role of assistant for this. Alternatively, a new role, break-out-leader must be created.

From the user-requirements in Section 10.3 and especially from Appendix D can be derived that it should be possible to free a number (or all) of floors of all control. This increases the consistency of the HCI. It also makes the old functionality in LearnLinc possible, with uncontrolled text chatting and uncontrolled breakout sessions. Especially the availability of an uncontrolled text floor can increase the awareness of the users (See Section 9.2.6).

The above-mentioned options to define floor control for each floor (or the absence of floor control) mean the HCI must provide that functionality at least during the setup of a teleconference session.

Appearance

LearnLinc looks up-to-date but some HCI-aspect might be improved.

It has a toolbar-like appearance. The toolbar is always visible so it can never be "lost" when a user has opened many windows. It is divided into several areas, each with different functionality, that have a fixed order. Unfortunately, the size of each of those areas can't be changed manually. It can only be minimized or maximized. In addition, the width and length of the toolbar can't be modified. It is advisable to give the user more freedom to arrange the application to his/her needs [15].

A number of areas from the toolbar can be undocked from the toolbar. In that case, the size is fully adjustable, but there is the danger of having to manage too many windows that might overlap each other (see Section 11.4.1). It would be advisable to let the whole toolbar expand and have an expanded docked mode available. This enables the windows to be resized as non-overlapping client areas within one container window. A good example of this is the Marratech teleconference system (see Section 11.4.1 and Appendix M)

Some integrated parts like the shared whiteboard have no toolbar-docking mode at all. They can only be used as a separate window. This makes usage of the interface somewhat inconsistent. GUI's should be as consistent as possible [15] and it is therefore advisable to make those parts also dockable in the toolbar. To have enough room on the toolbar it is advisable to combine this with the above-suggested expandability of the toolbar.

The attendee-list area of the toolbar has a raised-hand counter (see Appendix M). To combine this with the above-suggested functionality of a floor request list, it is suggested to make this into a separate raised hand area of the toolbar. This shows all raised hands, with optional reasons, numbered and in order of appearance. This instantly shows the total number of raised hands. When collapsed, this area shows only the raised hand count.

The option to give a reason for a request can best be given in the form of a popup and not using an extra field since this would clutter the user interface. Raising a hand is normally done by left-clicking the raise hand button (see Appendix M). The popup-window can best be invoked by right-clicking the raise hand button to activate a popup menu to be able to extend this functionality later. This has the advantage that it is consistent [15] with the suggestion above for request grant and refusal.

The web-based communications centre (see Section 9.5.6 and Appendix M) is the part of the HCI used to define many properties of a meeting. It seems therefore best to integrate the above-mentioned floor control definitions for a meeting into this part.

12 Elaborations on some design and implementation details

12.1 Scope

This chapter will only focus on the further elaboration of design and implementation issues of a small, defined part of the HCI of a new DTC system using current technologies. This will be done using rapid prototyping tools. It will result in a limited working prototype demonstrating some aspects of a HCI for a new DTC system.

12.2 Approach

The new elaboration of design and implementation issues will be severely limited to only the design of a user-list and a request list, with options for floor control of such a system, as seen from the perspective of a meeting controller who also controls all floors at once without making distinctions between different floors (See Chapter 3.4 for definitions). Participant actions will only be simulated locally. Only one request per participant is allowed and only floor control will be modelled. There will be no help functions available and no login facilities. No provisions will be made for users entering or leaving.

To arrive at the above, the relevant parts of the requirements for a DTC system as given in Section 10.3 will be used. With these requirements, new HCI specifications using state charts as recommended in Section 11.3 will be made. This will be done in Section 12.3.

Using the above specifications and recommendations as given in Chapter 11 a limited implementation will be given in Section 12.4. The implementation of this HCI-part based will be based on having only a client side, without having a server side available. On the client-side, AJAX, using the Tibco General design environment, as recommended in Section 11.5, will be used. There will be no provisions for handling events from the server-side except some simulated ones.

An elaboration on suggestions from Section 11.7 to improve a selected current teleconference application will be done in Section **Error! Reference source not found.**. This will be done using edited sequences of images of the HCI of that application. This provides enough expression to illustrate the desired improvements without an actual implementation.

12.3 HCI specifications

12.3.1 User requirements

To design the in Section 12.2 mentioned HCI parts of a DTC system, the user requirements as given in Section 10.3 must be reviewed to select only the necessary requirements.

Since the groupmeeting controller is also floor controller for all floors at once, there is no need for extra groupmeeting control functions. These can be fulfilled using only floor control functions. This restriction is comparable with the limited old requirements for a DTC system as given in Section 3.5 and the design limitations as given in Section 4.2.2, with the exception of not implementing options for user deletion. These requirements will therefore be used as a base to start from. Of these old requirements, only the requirements needed for viewing and managing the user-list of the meeting are needed.

The elaborated old requirements can be found as task analysis in Appendix D. Due to the above-mentioned restrictions, only the tasks of the floor controller for managing the floor access list need to be taken into account. Tasks belonging to other roles do not have to be considered. In addition, tasks of the floor controller for entering and leaving a floor and for local media usage or control are of no concern for this part of the report.

After taking the old task analysis from Appendix D and the removal of the above-mentioned tasks from that task analysis, the remaining tasks of the floor controller are:

Checks presence of floor participants
- **View the floor participant list.**

Removes floor participants
- **Chooses a participant from the floor list and delete participant.**

Manages floor
- **Looks at floor access request list for request-summary.**
- **Rejects floor access requests.**
- **Give floor access to that participant for one or more floors.**
- **Wait for participant to release floor access.**
- **Take back floor access from a participant for one or more floors.**
- **Take back floor access from all participants for all managed floors.**

This analysis needs to be adapted for this case. The floor controller is also groupmeeting controller and controls all floors. Therefore, the term "Controller" will be used from now on. Since all floors are controlled at once, floors are given a taken at once, without needing to select floors separately. Furthermore, in Section 10.3.3, an extra user requirement was recommended: to show request and grant indications for the floor controller also in the floor participant list. To follow this recommendation, the task list for the controller would need to be changed to include the resulting interaction within the tasks. Furthermore, there are no provisions for users entering or leaving the DTC system, so users can't be removed from a groupmeeting.

Checks presence of groupmeeting participants
View the groupmeeting participant list.

Manages floor
Looks at floor access request list for request-summary.
Looks at participants list for request-indication.
Rejects floor access requests.
Give floor access to that participant for all floors.
Wait for participant to release floor access.
Take back floor access from a participant for all floors.
Take back floor access from all participants for all floors.

12.3.2 Mapping from tasks to functionality

A mapping from these tasks to functionality the HCI has to offer must be made. This is done in the same way as was done in the first part of this report in Section 4.3 and in Appendix E. This is shown below.

Task analysis	**Functionality**
Checks presence of groupmeeting participants	Present groupmeeting participants list (show participants joining and leaving).
Removes groupmeeting participants	
Manages floor	
Looks for request-summary	Present floor-access request-list (show new requests).
Looks for request-indication	Present floor-access request indication in part.list.
Rejects floor access requests	Offer option to deny requests for floor access.
Gives floor access to participant for all floors.	Offer option to give floor access to participant.
Waits for participant to release floor access.	Show floor release indication.
Takes back floor access from that participant for all floors.	Offer option to grab floor access from participant.
Takes back floor access from all participants for all floors.	Offer option to grab floor access from all participants.

As can be seen, in accordance with the approach used in the first part of this report, the task description is already much refined, so the mapping of tasks to functionality has a 1:1 relation.

12.3.3 Modelling user interaction

Introduction

As recommended in Section 11.3 the user interaction of the HCI parts of the DTC system to be designed will be modelled using state charts. To do this, the relevant parts of the DTC system must be defined in terms of states and events. Events will be first be defined as user actions on a high level. A separate mapping from high-level user actions to lower level user actions will be provided.

For simplicity, this will first be done for only one participant and controller. Later this will be extended towards the final definitions and modelling for the relevant parts of the DTC system having more participants.

Unfortunately, the tool, selected in Section 9.2.3, Tau 4.2 from Telelogic that should be available here at computers of the CS department at the University is currently unavailable. This means the statecharts will be designed using only standard graphical office tools.

Defining states (one participant)

To model the interaction of the HCI functionality for a DTC system on the controller side in statecharts, as described in Section 12.3.2, the functionality of the HCI must be defined using states. To do this, the DTC HCI functionality was first defined for a DTC system supporting only one user and one controller. A first main decomposition of such a system using statecharts can be made using the orthogonal AND decomposition where a system must be in all of its AND-sub states [62].

This system was divided into 4 orthogonal groups:

- Participant
- Controller
- Request list
- Participant list entry indication

These are the most distinguishable objects for interaction with the limited DTC system. The participant and the controller are both different users that interact with the DTC system. Hereby the participant is only simulated locally (see Section 12.2), so no network effects need to be taken into account. The request list and the participant list entry indication both show part (or all) of the floor status information for a participant. The participants list is static since users can't be added or deleted (see Section 12.2). Therefore, only its dynamic part, the entry indication, is modelled. The rest of the HCI is only static and/or simulated and therefore does not have to be modelled.

Both user and controller may only take certain actions in certain states. They are therefore defined first. The HCI design must enforce these. In this design, the user is not a part of the GUI for the controller, but since actions from the user result in state changes for the controller HCI, it is still a part of the complete DTC system.

Since this first model only takes one participant in account, the request list top-states "has entry" and "has no entry" are the only ones needed.

In each of the AND-sub states, the system can be in only one state at a time. State changes occur because of events that take place. The main states of each group are as follows:

Participant:	has floor, has no floor.
Controller:	shares floor, shares no floor.
Request List:	has entry, has no entry.
Participants list entry indication:	attending, requesting floor, floor granted

Defining events (one participant)

Events cause state transitions to take place. In this case, the events are chosen to be abbreviations of actions made by the users on a high level of abstraction. This keeps the resulting statecharts clean and readable. These high-level user actions can later be mapped to user actions of a lower level like, for instance, pressing a key.

Users for the DTC system are the participant and the controller. The possible events these users can generate for the DTC system are:

Events	**High-level user actions**
P_F_Request:	Participant request floor from controller.
P_FR_Withdraw:	Participant withdraws floor request to controller.
P_F_Return:	Participant returns floor to controller.
C_FR_Denial:	Controller denies floor request from participant.
C_F_Grant:	Controller grants floor access to participant.
C_F_Grab:	Controller grabs floor access from participant.

Defining high level user actions (one participant)

Since of high level development environments (see Section 9.4) will be used that use standard widgets that perform standard functions, it is not needed on this level to describe how those user-actions are performed in extreme detail. It is best to provide a mapping of the above mentioned high-level user actions to more standard GUI user-actions. These standard user-actions require GUI functionalities that are standard for each development environment. These might differ somewhat between different development environments and Operating Systems, but are of less concern for modelling user interaction in the design phase. It is, for instance, not necessary to go into detail about how to make a selection in a list.

Next to GUI-button actions, also a keyboard mapping must be provided as recommended in Section 11.4.5. The button-action for granting a floor is in this design separated in two because button-icons and/or -text can be made dependable of the circumstances: It can be “grant floor” when no request has been made and “grant request” when a request has been made.

The user actions for the controller are:

High level user actions	**GUI user actions**
Controller denies floor request from participant.	Press ctrl+alt+“d” key or select “Deny request” button.
Controller grants floor to participant.	Press ctrl+alt+“g” key or select “Grant Request” or “Grant Floor” button.
Controller grabs floor from participant.	Press ctrl+alt+“f” key or select “Grab floor” button.

Statechart (one participant)

Using the above decomposition and events, a specification of the DTC system for only one participant using a statechart is constructed as shown below in Figure 53.

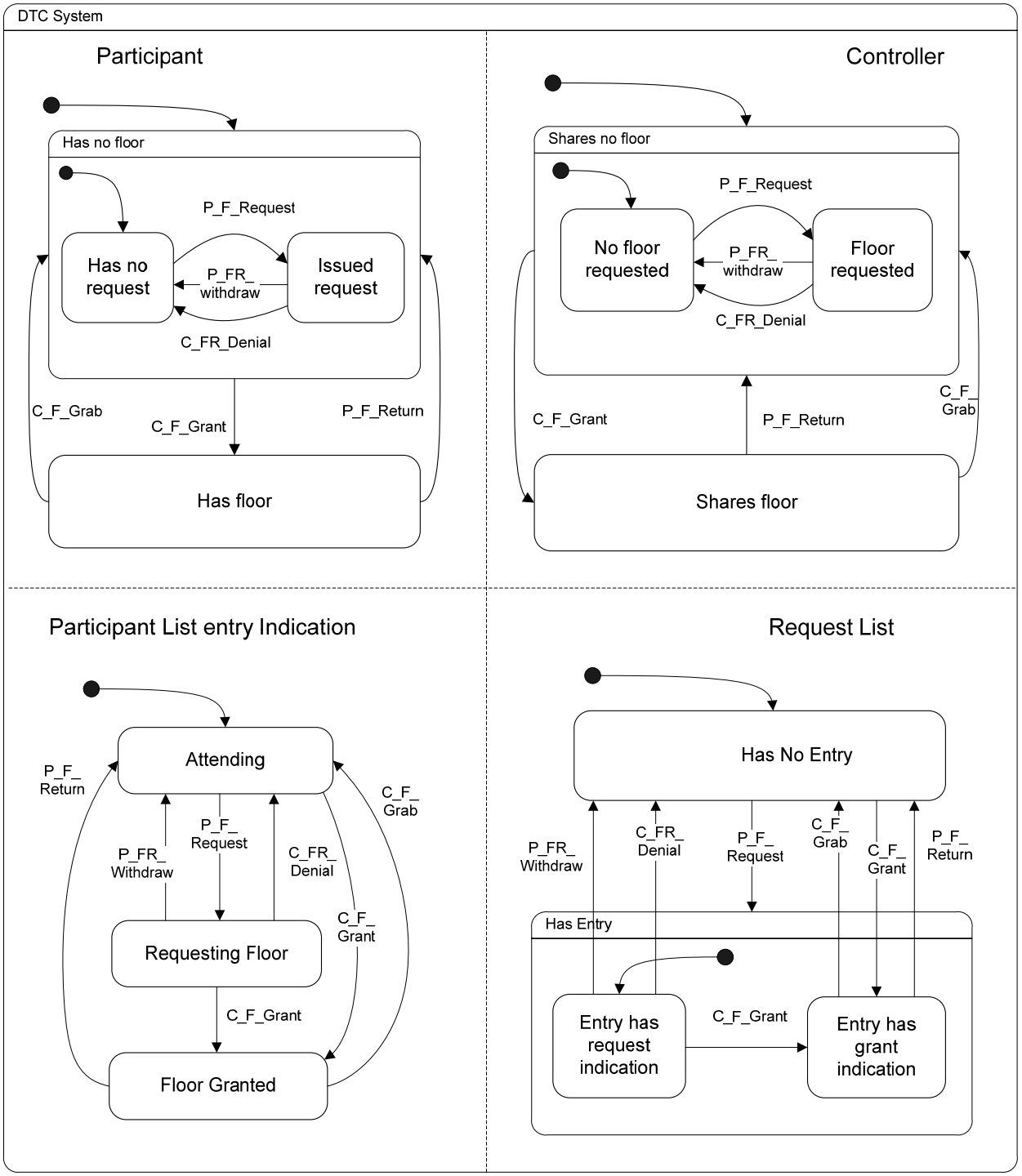


Figure 53: Statechart showing the most important elements of the DTC system concerning the controller, combined with one single participant.

Defining states (more participants)

To define the states for more than one participant, the 4 above defined AND sub states for one single participant must be repeated for every added participant. This is symbolized in statecharts with a "stacked" range of AND states. Each AND-state has a number attached to it "– n" that stand for participant nr n.

This also holds for the request list since only one entry per participant is allowed as indicated in Section 12.2. Therefore only two top-states per participant ("participant n has request entry" or "participant n has no request entry") are possible. This is much like the model for one single participant.

Defining events (more participants)

All events for the model with only one participant are extended with the number of participants by adding a number n, to model DTC system with more participants. Furthermore, there is one extra event to define, as according to the functionality as described in Section 12.3.2 there also must be a function to grab all floor access from all users at once.

Events	**High-level user actions**
P_F_Request n:	Participant n request floor from controller.
P_FR_Withdraw n:	Participant n withdraws floor request to controller.
P_F_Return n:	Participant n returns floor to controller.
C_FR_Denial n:	Controller denies floor request from participant n.
C_F_Grant n:	Controller grants floor to participant n.
C_F_Grab n:	Controller grabs floor from participant n.
C_F_Grab_all	Controller grabs floor from all participants

Defining high level user actions (more participants)

The high-level user actions for the controller are about the same as the ones for one participant. This is with the exception of the added number n and the new option to grab floors from all users at once.

The user actions for the controller are:

High level user actions	GUI user actions
Controller denies floor request from participant n.	Select entry in participant-list or request-list, next, press ctrl+alt+“d” key or select “Deny request” button.
Controller grants floor to participant n.	Select entry in participant-list or request-list, next, press ctrl+alt+“g” key or select “Grant Request” or “Grant Floor” button.
Controller grabs floor from participant n.	Select entry in participant-list or request-list, next, press ctrl+alt+“f” key or select “Grab floor” button.
Controller grabs floors from all participants.	Press ctrl+alt+”a” key or press “Grab All” button.

Statechart (one participant)

Using the above decomposition and events, a more final specification of the DTC system for more participants using a statechart is constructed as shown below in Figure 54.

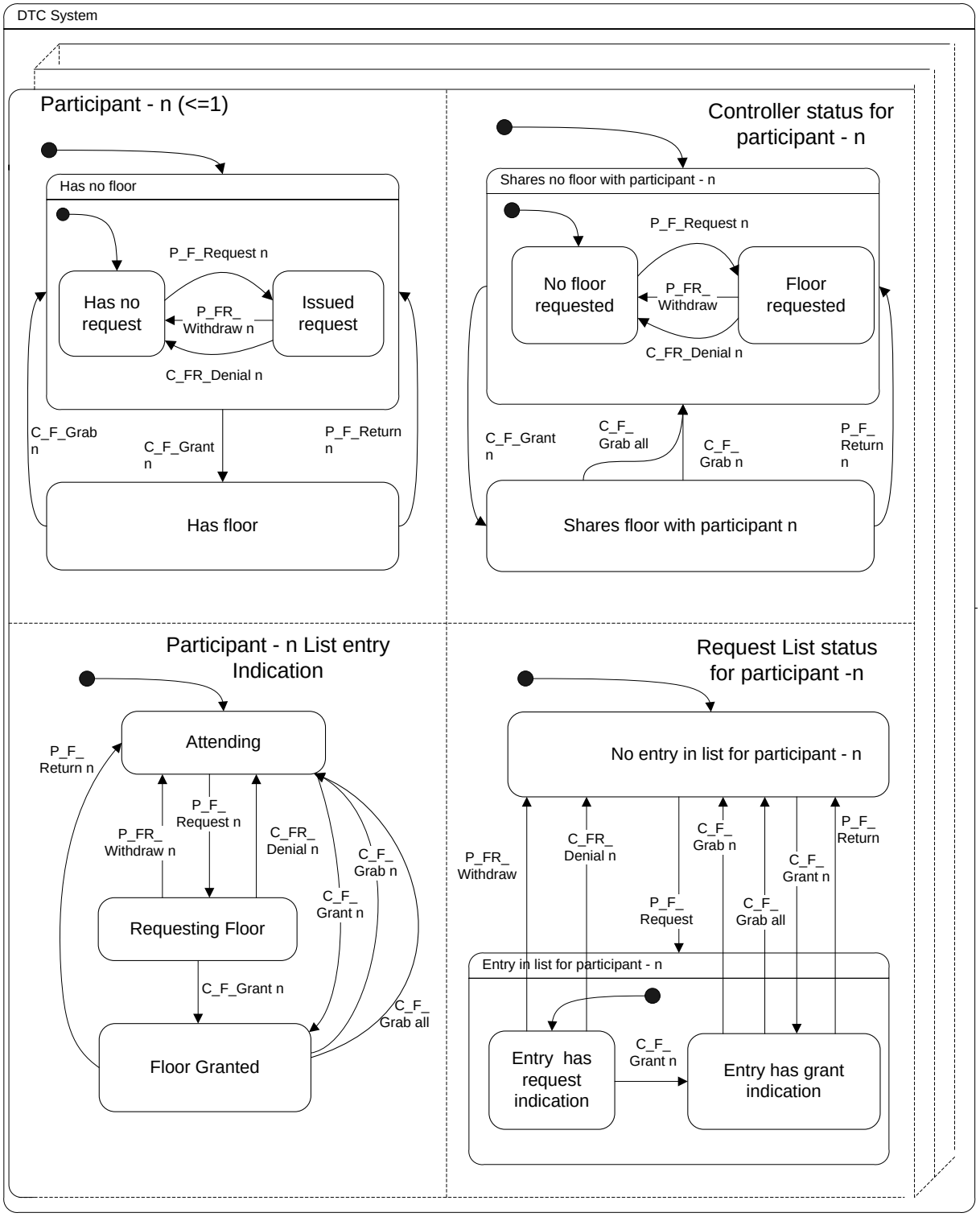


Figure 54: Statechart showing the most important elements of the DTC system concerning the controller for each participant.

12.4 Web based prototype using AJAX and Tibco General

12.4.1 General layout

As has been indicated in Section 12.2, Tibco General will be used to build a prototype. Tibco general is based on AJAX (see Section 9.4.2). The client side of an AJAX application normally communicates with a server. Here it is used in a stand-alone mode where all the necessary information is loaded at start-up from xml-files and is stored in a local data cache in XML format. Data can be read from and written to this cache without the need for a server synchronising that data in the background [63].

As been indicated in Section 12.2, a participant list and a request list with options for floor control will be designed. The functionality of those lists and the controls, including keyboard mappings are given in Section 12.3.3.

The GUI will be designed as one window, with multiple separated areas as recommended in Section 11.4.1. These contain the participants list, the request list and mock-ups of a text-chat area, a video area and a presentation area. Next to these, there is an area with menu-buttons to be able to simulate participant responses: requesting the floor, withdraw a request and returning a floor.

As is recommended in Section 11.4.2, Usage of colours will be kept to a minimum. Most colours used are shades of blue/gray. Only icons (see below) the video area and the presentation area (if the content has colour) stand out.

12.4.2 Buttons and icons

The buttons will be positioned where possible within the lists as recommended in Section 11.4.4. To do this, the buttons must be small. Therefore they will be designed as icons as recommended in Section 11.4.3. Tooltips will provide their function and key mappings in text. In addition, buttons outside the lists will be designed as icons to increase consistency and to maximize the space for other useful GUI-elements.

The graphics of these icons is designed to be consistent into families (see Section 9.2.5 and Section 11.4.3), representing floor access and requests.

 This is the basic icon for floor access, symbolizing a text balloon. Blue means a user has no speech rights. Green means a user has speech rights. Green is combined with a red cross symbolizing clicking on it means a cancellation of speech rights.

 This is the basic icon representing a request. Clicking on it grants a request. This icon with a red cross through it means a denial of a request.

Next to the basic families, each button type has been designed for animation, showing mouse-over events, pushdown events etc.

12.4.3 Internal design

Most the design is made directly using the editor of Tibco General. The main HCI functions for list handling are programmed in one package: “eg.dtc_hci_functions”. The listing of this can be found in Appendix N.

The lists conditionally show icon-buttons. To hide or show them, small format handlers needed to be written. These can also be viewed in Appendix N.

The user list and the request list are kept in cache XML format. The contents of these lists can be directly manipulated using JavaScript and the result of these manipulations can be viewed in the GUI-window after e refresh command for the corresponding graphical widgets. In the future, these XML-lists could also be synchronized with e server. This user list is initially retrieved from disk. See Appendix N for details.

12.4.4 Participant simulation

The right segment of the GUI screen is dedicated to simulate the actions of other participants besides the controller. It provides three pull down menus the each provide one functionality of the DTC system for all participants. The menu is labelled with the functions and when pulled down, the user can chose for which participant it applies. The functions are request floor, withdraw request, and return floor. Beneath the request floor function, a textbox is provided that can contain an optional request summary (or reason).

12.4.5 Graphical view

An example of how the resulting application looks like, with a few outstanding floor requests and floor grants is given below in Figure 55.

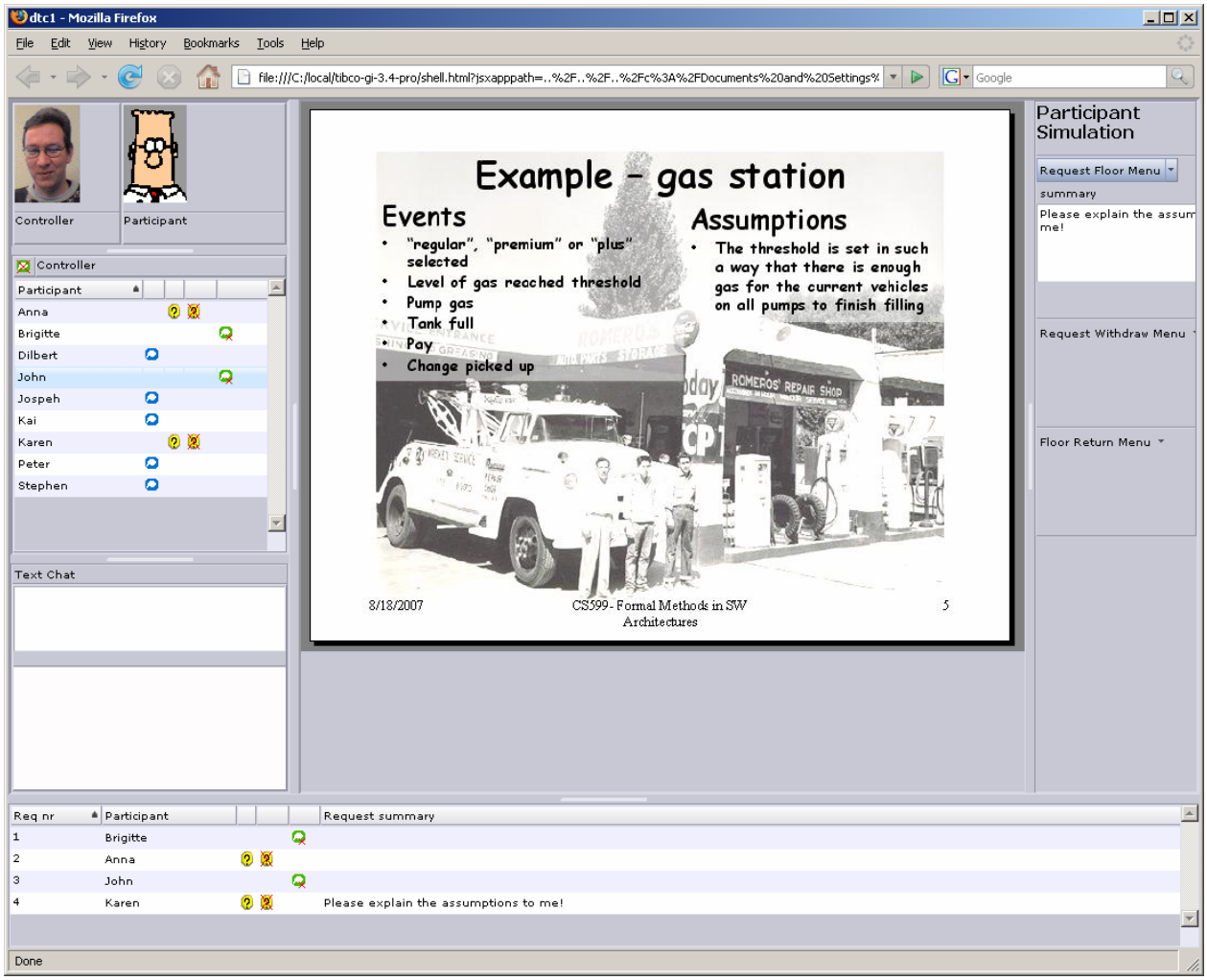


Figure 55: Example of the prototype of a new HCI for a DTC system.

13 Conclusions and recommendations

13.1 Conclusions

The old DTC HCI implementation looks dated and is not able to function fully anymore. However, as can be seen from the second part of this report, the basic design of the DTC system still stands. With a few small improvements, it can still be used for designing a current DTC system. As for the software architecture used for the old DTC system, this is also usable today. The new recommendations even arrive partly at the same software architecture. In this architecture, a GUI oriented system is used to design most of the HCI. This has to be combined with a layer of separate, specialized code for media streaming.

It is also clear that, when looking at current teleconference applications, that there is room for improvement of many of these applications, using the improved design of this report as a guideline.

The specification method using statecharts delivers sometimes rather trivial results when modelling direct manipulation interfaces where every function directly has its own button and/or key. It is however still very useful to model possible and allowed sequences of actions. Unfortunately it does not scale very well when it has to model 1:n or n:m relations.

Software design systems have improved enormously. Compared with TCL/TK and GuiBuilder, it is clear that a modern and powerful software design system like Tibco general has much more to offer. In hardly more than a week time, it was possible to create an acceptable looking GUI that also has some of the data-structure in background in place for bookkeeping of floor access and requests. These systems enable a software designer to accomplish much more in less time compared to 10 years ago.

13.2 Recommendations

There is still room for improvement of many of today's applications, using the improved design of this report as a guideline. It is therefore recommended to explore possibilities in that direction, handing recommendations of this report over to designers of these applications.

Because there is room for improvement in the area of teleconference systems specialized for educational purposes, it is recommended to explore options to continue design of a low-cost open source DTC system at the University of Twente.

Unfortunately, it was only possible at this moment to explore HCI design without actually confirming the working combination of AJAX technology with external Mpeg coding/encoding technology. When starting an implementation based on this report, it would be recommended, to first do research into actually combining these technologies in a working teleconference application.

As mentioned in Section 13.1, the statecharts modelling method does not scale very well when it has to model 1:n or n:m relations. It is however, an open method for which extensions have already been made. It is recommended to do more literature research to find extensions for statecharts that are better able to cope with this kind of relations.

When designing a HCI, it is recommended to do user-surveys and tests to find out which HCI functions are the most effective for users.

REFERENCES

[1] V.H.F.M. Korenromp, *On the Prototyping of a Multimedia Conferencing System for Desktop Tele-classroom Conference Application*, University of Twente, The Netherlands, 1995.

[2] A. Jongman, *On the Prototyping of Audio and Floor Control Protocols in a desktop Tele-classroom Conference Application*, University of Twente, The Netherlands, 1996

[3] ISO DIS 9241-11, *ergonomic requirements for office work with Visual Display Terminals: Part 11 guidance on Usability.*

[4] R. Huis in 't Veld, *Platinum Project, Conference Management Application Design,* CTIT, University of Twente, The Netherlands, 1996

[5] X.M.R. Wilhelm, *Logistics for the Off-line Communication in a Desk-top Tele-education environment,* University of Twente, The Netherlands, 1996

[6] Gary Perlman. *User Interface Development*, Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, U.S.A. , 1989

[7] Prof. M. Jakobsson, *Analysis and Design of Human Computer Interaction (lecture notes)*, University of Vaasa, Finland, 1996

[8] ir M.J. van Sinderen, ir A. Pras, *An introduction to computer networks (lecture notes),* University of Twente, The Netherlands, 1987

[9] J. Oosterhout, *The Relationship between Tcl/Tk and Java,* Sun microsystems, U.S.A., 1996

[10] Sun Microsystems, *The Java Tutorial,* Sun Microsystems, California, U.S.A., 1995

[11] B. Welsch, *Practical programming in Tcl and Tk,* Mountain View, U.S.A., 1995

[12] International Telecommunications Union (ITU), *Real time audio-visual control for multimedia conferencing (T130) draft*, 1996

[13] Marc Berenschot, *Object-oriented implementation of a DTC system in a Windows-NT environment*, 1997

[14] Michel Schudel, *Participant pictures in a desktop tele-classroom application*, Universit of Twente, 1998

[15] Shneiderman, Plaisant, *Designing the user interface (4th edition)*, 2005

[16] Preece, Rogers, Sharp, *Interaction Design - beyond human-cumputer interaction*, 2002

[17] R Rienks, *Meetings in Smart Environments,* University of Twente, draft 2007

[18] ISO/IEC JTC1/SC29/WG11, *MPEG-4 Overview - (V.21 – Jeju Version),* 2002

[19] Xu, G. Sugimoto, T. , *Rits Eye: a software-based system for real-time face detection andtracking using pan-tilt-zoom controllable camera*, Ritsumeikan University, Japan, 1998

[20] Ronald Poppe, Rutger Rienks, *Parlevison: Rapid Development of computer vision Application draft, webasite,* University of Twente, 2006

[21] Chris Aniszczyk, Eclipse *DeveloperWorks Article (www),* IBM, 2006

[22] Jesse James Garrett, *Ajax: A New Approach to Web Applications,* Adaptive Path, 2005

[23] Paulson, L.D., *Building rich web applications with Ajax,* Computer (IEEE), 2005

[24] Cynthia Calongne, Jeff Hiles, *Blended Realities: A Virtual Tour of Education in Second Life,* WSU Center for Tecaching and Learning, 2007

[25] Cheng-chao Su, *An open source platform for educators,* National Taiwan Junior College of Performing Arts, 2005

[26] S. Graf, B. List, *An evaluation of open source e-learning platforms stressing adaptation issues,* Vienna University of Technology, 2005

[27] Peter Wayner, *Top AJAX tools deliver rich GUI goodness,* Infoworld, 2006

[28] Frank Gilbane, *What is Content Management,* The Gilbane Report, Volume 8, 2000

[29] IBM HiPODS Latin America Team, *Web 2.0 -- Advanced AJAX Applications with the Google Web Toolkit,* IBM, 2007

[30] IBM, *Eclipse Platform Technical Overview,* IBM, 2006

[31] Gary Horen, *Exploring Ajax Runtime Offerings,* Dev2Dev website, 2006

[32] Project expert op afstand, *videoconferencening tips en scenario's,* Surfnet, 2006

[33] Second life WIKI, *Linux Viewers and 3D Acceleration*, Wikipedia, 2007

[34] Peter Wayner, S*urveying open-source AJAX toolkits*, Infoworld, 2006

[35] Andrey Filippov, *AJAX, LAMP, and liveDVD for a Linux-based camera*, Linuxdevices.com, Elphel, 2006

[36] Microsoft, *Using the Windows Media Player Control in a Web Page*, Microsoft MSDN, 2007

[37] Jacqueline Emigh, *New Flash player rises in the web-video market*, Computer, 2006

[38] Cristian Darie, Bogdan Brinzarea, Filip Cherecheş-Toşa, Mihai Bucica, *AJAX and PHP: Building Responsive Web Applications*, 2006

[39] Miguel de Icaza, Jaime Anguiano, Lluis Sánchez, Niel Bornstein., *Mono FAQ: Technical*, Mono-project, 2007

[40] ANP, *Second Life amper gebruikt*, Volkskrant, 2007

[41] Intel, *Moore's Law*, Intel, 2007

[42] George Lawton, *The Wild World of 3D Graphics Chips*, Computer, 2000

[43] James F. Kurose, Keith W. Ross, *Computer Networking, a top-down approach featuring the internet*, 2005

[44] Alex Karabuto, *Digest 2005: Hard Disk Drives and SATA/SAS Controllers*, Digit-Life.com, 2006

[45] David Fishman, *Digital Video Versus Analog Video*, Ezine articles, 2007

[46] Infotrends, *New InfoTrends Study Indicates Continued Growth in Photo Imaging Services*, Infotrends, 2007

[47] Benjamin J. Kadlec , Erik. O. D. Sevre, David A. Yuen, Xili L. Yang, Evan F. Bollig, Yunsong Wang, Gordon Erlebacher, Maxwell Rudolph, *Web-cam's Potential for Collaborative Activities in the Earth Sciences*, Visual Geosciences, 2004

[48] JeanPaul Saman, *Streaming networks with VLC*, NLUUG, 2006

[49] Paul Thurrott, *Windows Media Player 10 Review*, winsupersite.com, 2004

[50] Microsoft, *Windows Media Video 9 Series Codecs*, 2006

[51] Needforcontent.com, *Joomla review*, 2007

[52] BBC News, *Laptops set to out sell desktops*, 2007

[53] Computer Industry Almanac, *Smartphones to Outsell PDAs by 5:1 in 2006*, 2006

[54] Adama Brown, *Rumors of the Pocket PC's Death Are Premature*, Smartphone & Pocket PC magazine, 2006

[55] Hai Yan, *Multimedia Extensions in General-Purpose Microprocessors*, University of Connecticut, 2005

[56] Telelogic, *Telelogic Tau 4.2 User's Manual*, 2001

[57] Andy Cockburn, Bruce McKenzie, *Evaluating the effectiveness of spatial memory in 2D and 3D physical and virtual environments*, Conference on Human Factors in Computing Systems, 2002

[58] Jim Van Meggelen, Jared Smith, and Leif Madsen, *Asterisk, The Future of Telephony*, O'Reilly Media, 2005

[59] Paul Thurrott, *OneStat: Windows Continues to Dominate*, Windows IT Pro, 2006

[60] RealVNC, *VNC - How it works*, 2007

[61] Jonathan Kaye, David Castillo, *Flash MX for Interactive Simulation (chapter 7: Describing behaviour using statecharts)*, Delmar Learning, 2002

[62] David Harel, *Statecharts: a visual formalism for complex systems*, Science of Computer Programming, vol. 8, 1987

[63] Tibco, *Tibco General Interface: Getting Started, Tibco General Interface: Developer Guide*, *Tibco General Interface: Component Guide*, 2007

University of Twente
Enschede - The Netherlands

Design and Prototyping of a Human-Computer Interface for a Desktop Tele-classroom Conference Application

APPENDICES

Author:	Marcel van Bergen
Date:	August 30, 2007
University:	University of Twente, Netherlands
Department	Electrical Engineering, Mathematics & Computer Science Architectures and Services of Network Applications
Graduation Committee:	dr.ir. I.A. Widya dr.ir. B.J.F. van Beijnum dr. ir. D. Konstantas

APPENDIX A:
Excerpts from examined literature

V. Korenromp, On the prototyping of a Multimedia Conferencing System for Desktop Tele-classroom Conference Application [1]

The D-student report From Vincent contains definitions, examples of already existing tele-conferencing systems (IVS, MMCC and CU-Seeme) and a first design for a Desktop Tele-classroom Conference system.

It contains a suggestion for a number of future design-cycles and their user-requirements.
The user-requirements he suggested are:

For the first design-cycle:

> The teacher performing the lecture produces an information-stream consisting of audio and video that all students must be able to receive.
> The students may, individually, pose questions to the teacher, using audio, so there must be a possibility for the teacher to receive an audio or text-stream from a student.
> Furthermore, posing and answering questions in a controlled way means that there must be a token mechanism to give the teacher control over who may pose a question (i.e. only the one with a token).

For the second design cycle:

> Transmission of audio information from students question to all other students

For the third design cycle:

> Possibility of video stream from student to teacher.

For the fourth design cycle:

> Audio and video streams possible between all users.

These design cycles are mostly dependent on the capabilities of the network layer. These are mostly already implemented as the present session layer [2] and are therefore not important anymore to the design of the HCI.

After the definition of these user-requirements, Vincent defines a simple token mechanism, using TokenPlease Request / Indication, a TokenGiven Request / Indication and a TokenGrab Request / Indication / confirm primitives.

For his very first implementation he uses a modified version of a CU-SEEME reflector together with an adapted version of an example network-application for the SunVideo system. This implementation is command-line controlled and allows transmission of audio and video from one user, the teacher to a number of others, the students. It does not yet have transmission from students to teacher or any actual controlling token-mechanism.

ISO DIS 9241-11, ergonomic requirements for office work with Visual Display Terminals: Part 11 guidance on Usability [3]

This is a method for the specification of the usability of Human Computer Interfaces and can be used to design such a Human Computer Interface with respect to usability.

Usability of a HCI in this context means the assessment of the capability of a HCI to be used in a particular context of use. This context depends on the nature of use, the tasks the user has to perform and the environment.

This method consists of six steps.

- First, the user-context has to be established. This can be done by describing situations in which the HCI will be used. The most important result of this step is a description of all possible actors and their roles.
- Second, for each actor-role, a task-analysis must be made.
- Third, usability requirements must be defined. These can be defined by combining the task analysis with priorities, given by the environment.
- From this third step, the desired functionality's of the HCI can be derived. This consists of a set of functions that the HCI has to offer the user.
- In this step, the minimal hard- and software system demands are determined. These demands are imposed by the working-environment.
- The actual design of the HCI. By using the derived functionality of the HCI, determined in step 4, combined with the system demands, determined in step 5.

Furthermore, a few tips on evaluation and some simple examples are given.

Project Platinum, Conference Management Application Design [4]

This is a document that describes the design and partly implementation of a conference management system.

The following subjects are interesting in this design:

- Design of a Human Computer Interface in a number of steps
- Using scenario's to distinguish actor-roles
- Hierarchically organised conference
- Floors and floor-control

Design of a HCI

The design of the HCI is a good example of how to use the ISO-9241-11 guidelines [3].
It starts by defining an environment, the actors and their roles.
Once all actor-roles have been established, for each different actor-role, a goal- and task analysis is made.

Using scenario's to distinguish actor-roles

A scenario defines what role an actor within a certain global situation has. A different scenario can give another role to the same actor who therefore acts different than in another global situation.

The different actor-roles for different global situations are described with help of scenarios.
For each different global situation, a different scenario with different actor-roles and role-descriptions has been made.

This means that, with a suitable scenario and defined actor-roles, the HCI of a general teleconference system can become a specialised one, like that of a desktop tele-classroom Conference system.

Hierarchically organised conference

The conference that is described in this Platinum-document, are hierarchically organised. There is one conference, that can consist of a number of groupmeetings and each of them can have a number of floors. Each floor is a communication-medium, like a text, video, audio or a shared whiteboard channel that can both be read from and be transmitted to.

Figure 56 illustrates that.

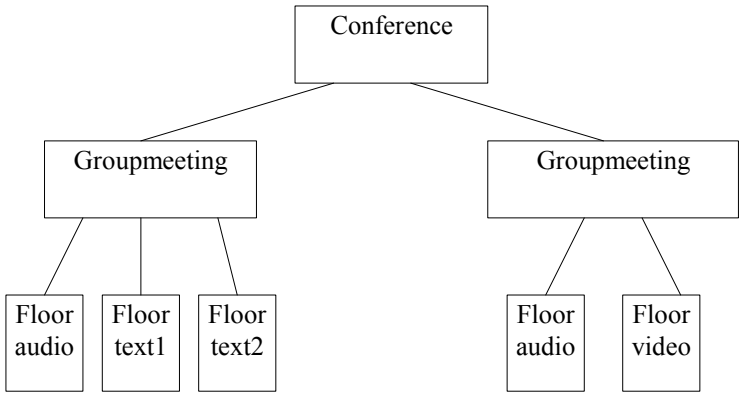


Figure 56

There can be any number of participants for each of the groupmeetings, floors and the conference, provided they are participants of the items that are hierarchically on top. Each of this groupmeetings, floor or conference is managed by only one controller.

Of course, a conference-participant can be controller for more than one item in more that one hierarchical layer at the same time.

Floors and floor control

In this Platinum document, control is exercised on basis of defined communication channels. Each channel can transfer one kind of medium, like audio, video, or text. This channel can be read from and/or written to and is called a floor. Each floor is controlled by a floor controller that controls write/transmit permission for that particular floor.

Each floor-participant can request a floor. This can be granted or rejected by the floor controller. The requesting participant will be notified of the grant/reject and after that, can take appropriate actions (e.g. start to speak). A floor-request can always be withdrawn by the requesting participant.

A floor controller can always “grab” a floor and thus deny further write-access to that floor for that participant.

Of course, a floor participant can always leave a floor on his own account, giving up any write permissions for that floor.

This is a more enhanced and universal form of communication-control, compared to the “token mechanism” for both audio and video as proposed in the report about the very first implementation of a Desktop Tele-classroom Conference system [1].

Gary Perlman. User Interface Development [6]

These lecture notes provide a lot of useful tips for designing and evaluating a user-interface.

Development / implementation

Advantage of rapid prototyping of user-interfaces:
- Design of user-interface is still very difficult and still lacks knowledge.
- Prototyping is necessary to evaluate and re-evaluate user-interfaces.

Advantage of user-interface design-tools:
- Increased consistency
- Eventually less programming effort necessary

Advantage of application-independent user-interfaces:
- Standardisation of user-interfaces for different applications.
- Easy adaptation of system to different types of input/output devices and different kinds of users.

User-guidance is very important. Not only in the form of a manual, but also as (error) messages, hypertext (help) and online tutorials. It must be consistent in contents and form.

Advantages of separation of the user interface itself and the user-guidance consisting out of on-line documentation (help), prompts and (error) messages:
- Easier to adapt to different languages.
- Consistency checking is easier.
- Text can be written by specialists.
- Adaptation to different users.

Mostly a user-interface is being used by both very experienced and inexperienced users. This must be taken into account.

Dialogue Types for User Interfaces:
- Questions & answers
- Forms
- Menu selection
- (Function) keys
- Command languages,
- Query languages
- Graphic interaction.

For not to experienced users, questions & answers, forms, menu-selection and graphical interaction are the best methods. For experienced users, (function) keys, command languages, query languages and graphical interaction are the best methods.

Customisation can be used to adapt dialogue types and user guidance to experienced or inexperienced users.

Evaluation

Evaluation of user-interfaces: Collect both user-patterns (more objective) and subjective ratings. Analyse with care.

Performance measures include:

- Error rates on selected tasks or over specified periods of use
- Learning time to be able to answer selected questions or perform certain tasks without referring to a manual
- Task completion time for selected tasks
- Amount of work done in a fixed time
- The location and duration of eye fixations
- Number of references to a manual.

Subjective impression measures usually are ratings and include:

- Rated aesthetics
- Rated ease of learning
- Rated throughput
- Stated decision to purchase.

Composite measures are usually weighted averages of any of the above measures.

Analyses of protocol (observe user, maybe let user explain what he/she is doing and thinking) is difficult and needs expertise.

Automatically monitoring of use (in a log-file) of user-interface can give information about problems and about which parts of the system are mostly used. This last can be taken into account for optimisation.

Xanter Wilhelm, Logistics for the Off-line Communication in a Desktop Tele-education Environment. [5]

Xanther describes in his D-student report amongst others an administration system for college registration, college schedules (both teachers and students) and grades awarding.

The different elements used there can be used as a guideline for the different elements and relations that can be used in a definition for a database that can be used within a Desktop Tele-classroom Conference system.
The advantage of such an approach is that the administration system as designed by Xanther and a future Desktop Tele-classroom Conference can easily work together with a minimum of changes.

The most important part is his description of the used data-structures in chapter 5 because part of these can be used (in an adapted form) by the Desktop Tele-classroom Conference system.
These data structures are divided in a static part that contains things like id-numbers, addresses, names and a dynamic part that contains course-logistical information. All dynamic data for teachers, students and courses is stored per college-period. Since this is of no consequence for this system, because there is only one relevant college period at a time, this college-period will not be taken into account. Also static stored data like grades, that are irrelevant for a DTC system, will not be taken into account. Data like photo's and e-mail addresses can be relevant for an extended Desktop Tele Classroom system, but will at the moment be ignored for simplicity.

First, the data elements that can be important for the DTC system are summarised.

The main elements, important for this DTC-application, are:

- Teacher/Student ID number
- Teacher/Student name
- Teacher/Student password
- Course ID number
- Course Name

Their relations are defined in tuples as follows:

- To determine the correct identity (static)

  Teacher/student ID number + name + (password)

- To determine the name of a course (static):

  College ID number + College Name

- To determine what course is being given, by whom, and who may join it (dynamic):

  College ID number + Teacher ID + List of student ID's

Alex Jongman, On the Prototyping of Audio and Floor Control Protocols in a Desktop Tele-classroom Conference Application. [2]

Short description of functionality of the underlying session-layer that will be used by the HCI.

The session layer as designed and implemented by Alex now permits the following functions:
- Floor control for different media (when controllers are known).
- Full duplex audio, video and text transmissions.
- Local media control for audio/video (audio volume, video size, coding).

As this layer itself presents audio and video to the user because of performance reasons, it even implements parts above the application layer, according to the ISO-OSI reference model [8]. Still, in this document, this layer will be called a "session"-layer. This is because it does not yet provide means to translate data between different Desktop Tele-classroom-Conference formats (Presentation layer, according to the ISO-OSI reference model [8]) and much of the resource-management is done by the HCI.

Details

The network-layer uses a CU-SeeMe reflector like in the very first implementation of the Desktop Tele-classroom system [1]. This reflector must be started separately. The network layer runs on Sun workstations under Sun Solaris and makes use of Sun video-digitising hardware and (optional) ATM network cards.

This network-layer is aware of media (floors) and permits a number of primitives that provide extended floorcontrol over those. This floor control is comparable with the floor control as presented in the Platinum documents above.

This layer implements a very complete form of media control. This is called floor control and lets users control each defined medium separately. Even the controller can differ per medium.

Floorcontrol-primitives and direct control for media like audio and video are separated, so the HCI itself can decide what has to be done exactly with those media. This leaves amongst others the possibility open to "override" received floor control primitives.

The local controls for audio are only partly implemented. These must be extended by the HCI for now.

Local controls for video are present, but should be used with care. The HCI must take care of that.

Once a video stream has been established, the network-layer directly places a video-image on the Xwindows desktop, without any direct control of a HCI.

The functionality of the session layer is good enough to implement a floor control over different, already existing and defined floors with a known floor controller. A floor controller can't be changed dynamically. On the next page follows a detailed description of the all service primitives provided by the session layer.

Service primitives for floor control:

Called by HCI:

Text_stream_req [channelID] [text,255]
Broadcast [text] as text to all
FP_Tokenplease_req [medium,19] [dest_IPaddress] [userdata,80]
Ask for a token to [dest_Ipaddress] with [userdata] as message
FP_Withdraw_req [medium,19] [dest_IPaddress]
Wiskdraw a tokenrequest
Tokengive_req [medium,19] [dest_IPaddress]
Give a token for access to [medium] to [dest_IPaddress]
FC_Reject_req [medium,19] [dest_IPaddress] [userdata,80]
Refuse a tokenreqeust for [medium] to [dest_Ipaddress] with [userdata] as message
FC_Tokengrab_req [medium,19] [dest_IPaddress]
Grab a token for [medium] for [dest_Ipaddress]
FC_Reset_req [medium,19]
Reset floor (grab all tokens) for [medium]

Delivered to HCI:

text_stream_ind [channelID] [source_IPaddress] [text,255]
Text reception of [text] from [source IPaddress] for [channel]
FP_Tokenplease_ind [media,19] [source_IPaddress] [userdata,80]
Request for token from [source IPaddress] for [media] with [userdata] as message.
FP_Withdraw_ind [media,19] [source_IPaddress]
Request for token from [source IPaddress] withdrawn for [media].
Tokengive_ind [media,19] [source_IPaddress]
Token received for [media] from [source IPaddress]
FC_Reject_ind [media,19] [source_IPaddress] [userdata,80]
Request rejected for [media] from [IPaddress] with [userdata] as message
FC_Tokengrab_ind [media,19] [source_IPaddress]
Token grabbed for [media] by [source IPaddress]
FC_Reset_ind [media,19] [source_IPaddress]
Floor reset for [media] (=al tokens grabbed) by [source IPaddress]
Adding [IPaddress] [Name,19]
[Name] at [Ipaddress] joins the conference
Deleting [IPaddress]
[IPaddress] leaves the conference. Video reception is stopped for [IPaddress]

Audio Control Commands:

Volume control	
PVolume [value]	*Set volume for audio playing*
RVolume [value]	*Set volume for audio recording*
set_stream_volume [IPaddress] [factor]	S*et relative volume for one audio source*
Reception control	
enable_read_stream	*Enable audio reception*
disable_read_stream	*Disable audio reception*
Transmission control	
audio_stream_enabled	*Enable audio transmission*
audio_stream_disabled	*Disable audio transmission*

Video Control Commands:

video_stream_enabled	*Enable video out transmission*
video_stream_disabled	*Disable video out transmission*
display_video_stream [IPaddress]	*Show video transmissions from [Ipaddress]*
hide_video_stream [IPaddress]	*Hide video transmissions from [IPaddress]*

The following commands MUST be executed before a video transmission/reception is enabled. When a setting has to be changed later, the transmission or reception of that image must first be disabled, and then the settings can be changed. After that, the transmission/reception can be re-enabled.

set_video_in_encoding [CellB|Jpeg|UYVY]	*Set video encoding for transmission*
set_video_out_decoding [IPaddress] [CellB|Jpeg|UYVY]	*Set video decoding for reception from [IPaddress]*
set_video_in_preview_on	*Set transmission preview on*
set_video_in_preview_off	*Set transmission preview off*
set_video_in_scale [scalefator]	*Set transmission scale (0=large,1=normal, 2=small)*
set_video_in_xy [xpos] [ypos]	*Set [xpos],[ypos] for transmission preview*
set_video_out_xy [IPaddress] [xpos] [ypos]	*Set [xpos],[ypos] for videoreception from [IPaddress]*

General control commands:

quit	*Leave conference, stop network layer*

Database commands:

who	*Give list of conference-members with setting-info (for use without HCI)*
status	*Give local status (for use without HCI)*

Uses of service primitives

In case of audio and or video, the user-application just switches transmission on or off with video_stream_enabled, video_stream_disabled, audio_stream_enabled, or audio_stream_disabled, according to permissions, received with a Tokengive_req/ind, a FC_Tokengrab_req/ind or a FC_Reset_req/ind.

In case of text, the application must use the service primitives to decide whether or not to allow a text to be transmitted with a text_stream_req.

Whiteboards are not yet considered, but might use the text transfer service primitives as well.

With the service primitives FP_Tokenplease_req/ind, FP_Withdraw_req/ind, Tokengive_req/ind, and FC_Reject_req/ind a list-system can be implemented that shows requests and makes it possible for a floor participant to withdraw a request, or for a floor manager to grant or refuse a request.

The service primitives do not provide options to delete a user from one or more floors.

Marc Berenschot, Object-oriented implementation of a DTC system in a Windows-NT environment, 1997 [13]

Marc Berenschot has made an implementation of the session layer that has the same service primitives as the above described session layer for a DTC system by Alex Jongman [2] for Microsoft Windows NT (version 3.51 and 4.0). It uses basically the same inter process communication mechanism as Alex version. It uses a pipe to the starting process that can be used to exchange service primitives with parameters. The service primitives are the same as in the implementation of Alex Jongman [2] and it can connect to the same adapted CU-Seeme reflector

This meant the HCI had to be adapted to be used in a windows environment. For a start, the way the session-layer program is actually started has to be adapted to a windows environment. Furthermore, special time-out functions were needed to prevent (in most cases) the crashing of the session-layer and/or to restart the session-layer when a crash does occur.

General audio settings are not (yet) possible using available utility programs, so these functions return without doing anything in a windows environment.

Michel Schudel, Participant pictures in a desktop tele-classroom application, 1998 [14]

Image HCI code interface

This document describes how to use the Image HCI interface into the main HCI for the DTC system.

The main HCI must, at some point in the code, include the line

```
source "ms_ihci.tcl"
```

This will include the image HCI into the main HCI. The file `ihci.tcl` will make sure all the modules needed for the image HCI are included as well.

Exported functions (first version: Build 425):

The image HCI delivers a number of procedures that can be used by the main HCI to visually represent users that are added, deleted, are talking or are silent.

`proc ms_ihci:adduserbynumber {userid}`

This procedure visually represents a user who has just entered the conference.

`userid`	a decimal number that represents the user. This number is unique, and used as a search key in the database.

`proc ms_ihci:deleteuserbynumber {userid}`

This procedure visually represents a user who has just left the conference.

`userid`	a decimal number that represents the user. This number is unique, and used as a search key in the database.

`proc ms_ihci:adduserbyname {name}`

This procedure visually represents a user who has just entered the conference.

`name`	a text string that represents the name of the user. This string is used as a search key in the database. *Warning: this assumes that the name string is unique, and the function consequently will retrieve the first name in the database that matches the string.*

`proc ms_ihci:deleteuserbyname {name}`

This procedure visually represents a user who has just left the conference.

`name`	a text string that represents the name of the user. This string is used as a search key in the database. *Warning: this assumes that the name string is unique, and the function consequently will retrieve the first name in the database that matches the string.*

`proc ms_ihci:talkuserbynumber {userid}`

This procedure visually represents a user who has started 'talking', or, more generally, a user that has

received a token for floor access.

`userid` a decimal number that represents the user.
This number is unique, and used as a search key in the database.

`proc ms_ihci:untalkuserbynumber {userid}`

This procedure visually represents a user who stopped 'talking', or, more generally, a user that has lost his token for floor access.

`userid` a decimal number that represents the user.
This number is unique, and used as a search key in the database.

`proc ms_ihci:talkuserbyname {name}`

This procedure visually represents a user who has started 'talking', or, more generally, a user that has received a token for floor access.

`name` a text string that represents the name of the user. This string is used as a search key in the database. *Warning: this assumes that the name string is unique, and the function consequently will retrieve the first name in the database that matches the string.*

`proc ms_ihci:untalkuserbyname {name}`

This procedure visually represents a user who stopped 'talking', or, more generally, a user that has lost his token for floor access.

`name` a text string that represents the name of the user. This string is used as a search key in the database. *Warning: this assumes that the name string is unique, and the function consequently will retrieve the first name in the database that matches the string.*

`proc ms_ihci:reset`

This procedure resets all users to their original state, e.g. not 'talking'. Typically called when the main HCI issues a `FP_Reset_req`.

`proc ms_ihci:init_image_hci`

This procedure is called by the main HCI to initialise the image HCI. This will read the proper database files, initialise widgets, set event bindings, make the correct widgets visible, etc. Typically called somewhere at main HCI initialisation time.

Exported functions (second version: Build 445):

ms_ihci:command command user how

command = (see below)

adding

This procedure visually represents a user who has just entered the conference.

deleting
This procedure visually represents a user who has just left the conference.

tokengive
This procedure visually represents a user who has started 'talking', or, more generally, a user that has received a token for floor access.

tokenreject
This procedure visually represent a token rejected.

tokenplease
This procedure visually represent a token requested.

tokenwithdraw
This procedure visually represent a token withdrawn.

tokengrab
This procedure visually represent a token grabbed.

reset
This procedure resets all users to their original state, e.g. not 'talking'.

hide
This procedure is called to hide the image-HCI.

show
This procedure is called to show the image-HCI.

init
This procedure is called to initialise the image HCI.

destroy
This procedure is called to destroy all windows from the image HCI.

user = a name or a number, depending on the method chosen (=how)
how = byname or bynumber

Notes on variable use and name clashing

Data entities are defined as follows in the image HCI. The 'x' and 'y' parts stand for arbitrary names.

Toplevel window named x	`.ihci_x`
Widget named y in toplevel window x	`.ihci_x.y`
Global variables	`ms_xxxx`
Proc x in module y	`ms_y:x`

Build 445 has added functionality but unfortunately a lot of errors. The interface of build 425 has been adjusted to also accept the interface commands of build 445. For now build 425 with an adapted interface will be used.

International Telecommunications Union (ITU), Real time audio-visual control for multimedia conferencing (T130) draft, 1996 [12]

Within the drafts for a teleconference standard architecture [12], the network functionality is similar to the one used here [2]. The conference model is somewhat different but similar to the floor control mechanism described in Alex's report [2] and platinum [4]. The division between floor controller and participant is made less strict in this draft than will be in the DTC system (It is best compared with the presenter and audience role).

Shneiderman, Plaisant, Designing the user interface (4th edition), 2005 [15]

This book presents modern theories and practices to design interfaces for interactive systems..

The first chapter introduces the UI design and its goals. It shows examples of interactive systems, and elaborates on usability of interactive systems, usability requirements, usability goals and measures and different motivations to enhance usability.

The second chapter introduces a number of general guidelines, principles and theories for designing good user interfaces.

A number of other general principles for designing user interfaces are:

- A designer must know for what kind of user the system will be designed. For instance the skill levels of potential users must be determined. Are you designing mostly for experienced users that will use an application intensively and probably first get a thorough training or does the application first of all need to cater for casual users that do have to find their way quickly without a steep learning curve.
- The tasks to be carried out and their frequencies need to be identified. This not only helps to specify all of the functionality of a user interface, but can assist in deciding what tasks not to support to prevent a cluttered user interface supplying an overload of tasks that users do not need. Furthermore this can assist in determining what actions need to be atomic and what actions can be composites of other actions.
- An interaction style must be chosen. Some interaction styles are: direct manipulation (manipulating or activating visual representations of objects or tasks), menu selection, form fill-ins, command language and natural language. Mostly there is not one correct choice, but a user interface will consist of a blend of a few of those styles, directed at different levels of functionality and user-experience.
- Always preserve some means of human control. As the pressure for productivity grows and procedures around a computer system become more standardized, many routine tasks for such a system will be increasingly simplified and automated. This is beneficial for users since it helps users to avoid tedious and error prone tasks. Still those tasks must have predictable and controllable interfaces to preserving human supervisor control. This is very important since the world is an open system where unpredictable events can always occur and humans can respond to these events. In contrast, a computer is a closed system: it can only accommodate for a number of predictable events. Human judgement is sometimes needed to increase safety, to avoid failures and to increase quality.

As most important of general principles of user interface design, eight golden rules of interface are listed:

- Strive for consistency. E.g. always use the same key-combination for the same function, always place a button or menu-item with the same function at the same location, always give the same function the same name etc.
- Cater for universal usability. Try to implement for needs of different users and design for plasticity. Try for the application to be usable for novices and experts alike. Let an application be usable for users of different age and abilities and be applicable to diverse technologies.

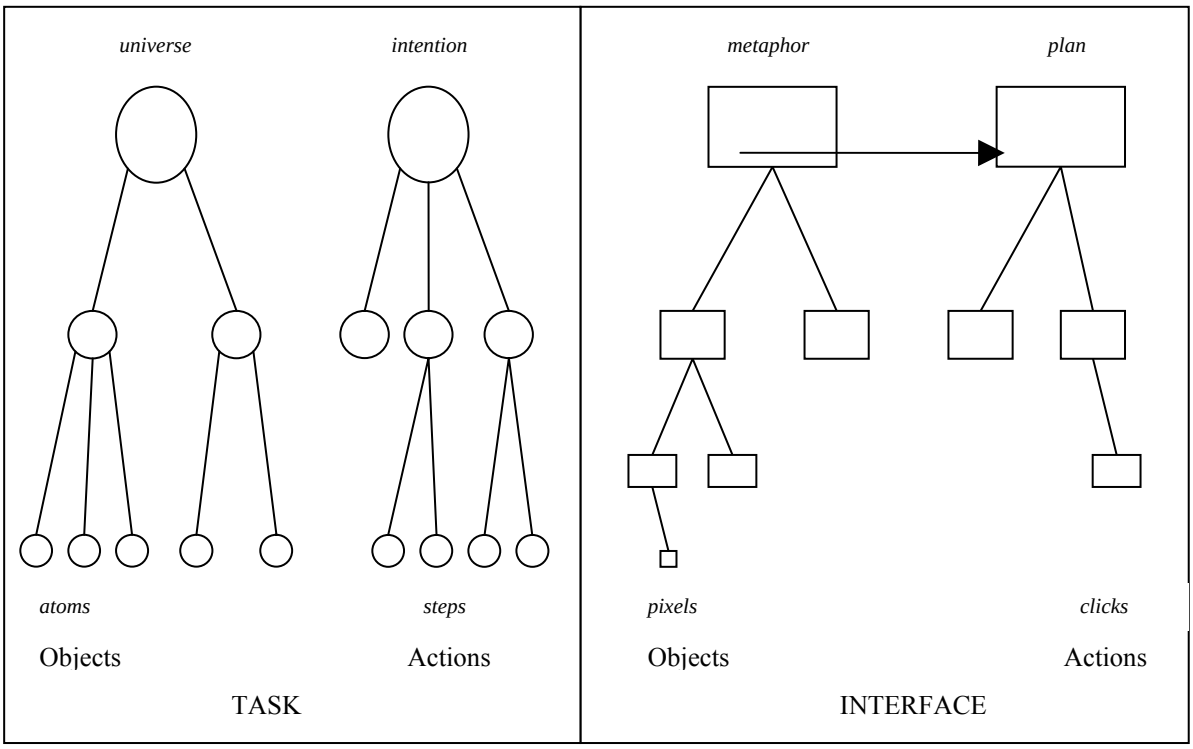
- Offer informative feedback. For every user action, there should be system feedback. For frequent and minor actions, the response can be modest whereas for infrequent and major actions, a more substantial response is required.
- Design dialogs to yield closure. Sequences of actions should have a clear beginning and a clear end and should offer informative feedback at the end of a sequence. This gives users a sense of accomplishment and relief, allowing them to drop contingency plans for that sequence and allowing them to prepare for a next sequence of actions.
- Prevent errors. As much as possible, a system should be designed so users can't make serious errors. For example, menu-items that are not appropriate should be greyed out and users should not be able to enter alphanumerical characters in a numeric entry field.
- Permit easy reversal of actions. As much as possible, actions should be reversible. This relieves anxiety since the user knows errors can be undone. Furthermore, this encourages exploration of the user interface.
- Support internal locus of control. Users strongly desire that they are in charge of the user interface and that it responds to their actions. Try, for instance, to avoid surprising interface actions, tedious sequences of data entries. A large part of this principle can be fulfilled by avoiding a-causality and making users the initiators of actions.
- Reduce short-term memory load. An average person can only remember seven plus or minus 2 things for a short time. Displays should be kept simple; multiple page displays should be consolidated and, windows motion frequencies should be reduced. When not avoidable, the use of codes, mnemonics and sequences of actions requires the users to have enough training-time available and access to online help explaining those.

A number of more specific guidelines for organizing displays are listed below:

- Present information in an efficient way to the user (make labels comprehensive, organize data in appropriate columns adapting properties like justification to the type of data presented).
- Be consistent in labelling and graphic conventions
- Standardize abbreviations
- Use consistent formatting in all displays
- Present data only if it aids the user
- Present information graphically where appropriate
- Present digital values only when knowledge of those values is necessary and useful
- Use high resolution displays
- Design displays in monochromatic form using spacing and arrangement for organisation and add colour judiciously where it will aid the user
- Involve end users in the development of new displays and procedures
- Minimize input actions needed by the user
- Provide consistent possibilities for actions and give consistent responses
- Maintain compatibility between data entry and data display
- Where possible, and if not in violation with consistency rules, provide flexibility for user control of information display.

A number of theories and models are presented but most of those have a limited use. E.g. GOMS and its derivatives model mostly actions of experienced users. Action grammars can be used to check consistency of the user interface, but this method might be too rigid in a number of cases. The most important model presented in this chapter is the Object-Action Interface Model (OAI). This model first identifies a hierarchy of objects, decomposed finally in so called atoms and intentions, decomposed into intermediate goals and finally into steps of a task in the real world. This is then put into a conceptual model of the interface containing a hierarchy of metaphors for the objects of the real world that are decomposed finally into so called pixels. The intentions in

the real world are mapped on plans that are decomposed into intermediate actions that are finally decomposed into keystrokes and clicks.



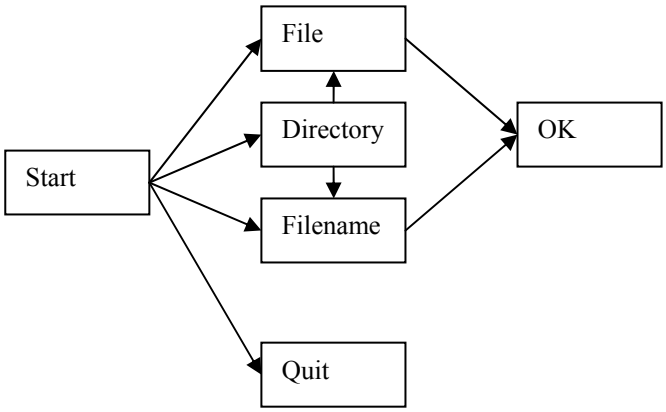
The OAI-model

Chapter 3 elaborates on managing of the design process. In a good design process one should use guideline documents that are living texts that can be adapted during the design process. These start with a set of working guidelines, derived from designing principles and guidelines as mentioned in chapter 2, 6 and 7, but can also contain subjects about design decision policies, education of users, etc. Next, one should use good software tools to design and implement user interfaces. One should use expert reviews and/or usability testing for evaluation. Evaluation of interface designs is separately treated in the fourth chapter.

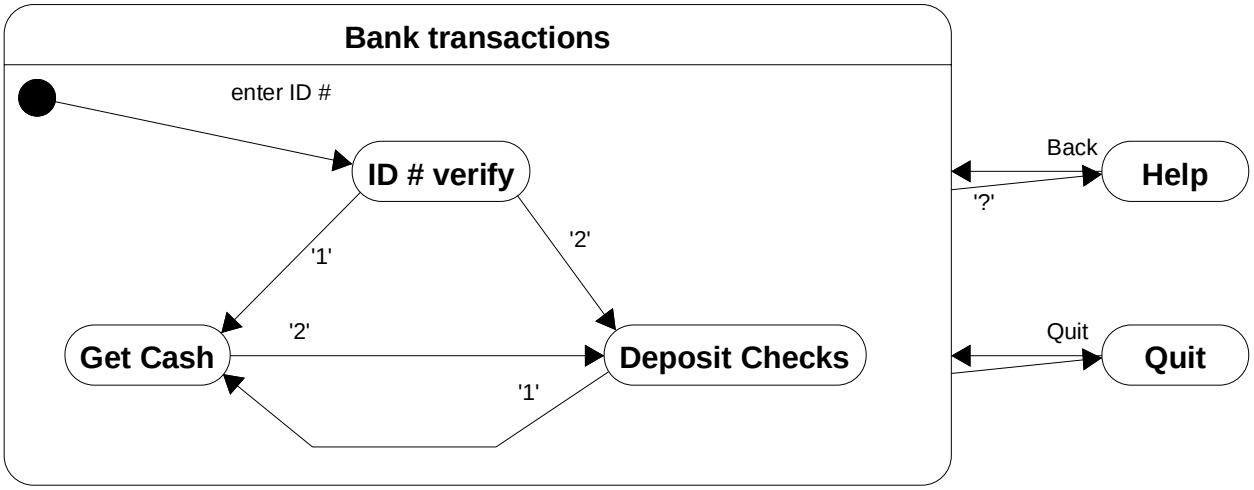
The fourth chapter is about Testing. It details how to do expert reviews (heuristic evaluation, Guidelines review, consistency inspection, cognitive walkthrough, formal usability inspection), how perform usability testing mostly using a testing laboratory (paper mock-ups, discount usability testing, competitive usability testing, universal usability testing, remote usability testing, field tests and portable labs). It also treats acceptance tests and evaluation during normal use. It shows what surveying instruments can be used.

The 5th chapter presents a number of available specification methods to be used in user interface design. A number of modelling and specification tools are treated like grammars, menu-selection trees and dialog-box trees, transition diagrams and state charts. Grammars are not very suitable for a modern interactive user interface since they have difficulty coping with 2d information styles like most modern user interfaces today have. Menu trees and dialog trees are quite powerful, and can show both high level relationships and low level details of a system and allow for consistency checks. A drawback of trees is that they become difficult to handle with large systems. Another drawback is that modern direct manipulation interfaces do not only have menus to control the system. A transition diagram exists of a set of nodes that represent the system

states and a set of directed links between the nodes that represent possible transitions. It is possible to label the nodes with transition frequencies when there are know. Transition diagrams have the advantage that they can be converted almost automatically in finite-state automata, a well known phenomenon in computer science. Several properties, like reachability can be verified automatically



Statecharts do not have the above disadvantages of transition diagrams. They have a grouping feature making it easier to handle complex systems and extensions are available to model concurrency, external interrupts, dataflow etc. The Unified Modelling Language (UML) also uses state charts to specify the behaviour of general programs, so its support is growing and several tools are available to assist drawing state charts.



Statechart example of a simplified bank transaction system

Next, it presents a number of software tools, available at the time of the last revision of this book. Those include Flash, Microsoft Visual Basic.NET, Borland Jbuilder, ILOG, Motiv, Java, Javascript, Tcl/TK, Perl/TK, Python/TK, Apple Hypercard and Labview. Tcl/TK is mentioned as the first scripting language that was good at prototyping user interfaces and high level in defining graphical elements of user interfaces.

Chapter 6 presents modern interaction styles based mostly on direct manipulation that are possible with present technology. It sketches the history of direct manipulation by looking at word processors. In the early 1980's, text editing was done with line oriented command languages. Next there came full-page display editors that allowed users to edit a whole page of

text using a cursor to move around. These evolved further and in the early 1990's the What You See Is What You Get (WYSYWYG) editors in which text on screen is shown in a 2 dimensional interface as it will print and objects in the text can be directly selected, dragged etc.

Functions are selectable by menu's, but also by clicking on icons that symbolize those functions. Icons have the advantage that they can be smaller then corresponding buttons with text. This means they can be put nearer to the objects they are meant to manipulate, if applicable. At the same time they can stand out more from the environment.

Direct manipulation is nowadays the standard user interface for many programs (Operating system shells, word processors, spreadsheets, games, Computer Aided Design (CAD), etc.

Direct manipulation has the advantage that it is transparent for many users. It makes the users feel they are involved directly with a world of objects and not with an intermediary. This is also shown when looking at the OAI-model in which the elements of the task block can be projected almost 1:1 on elements of the interface block. It might still require users to have a substantial knowledge of the tasks, but once a user has acquired this, there is only a modest amount of interface knowledge required.

Drawbacks of direct manipulation are that it is more difficult for vision-impaired users to handle, that it consumes valuable screen-space, that it requires users to learn the meaning of visual representations and that those can be misleading. A last problem is that many direct manipulation interfaces frequently requires users to use a mouse and this might cost an experienced user more time that it takes to type in commands at a keyboard.

As for the usage of icons, a number of tips are given:
- Limit the number of different icons
- Make the icon stand out from the background
- Be careful with 3D icons, they are catchy, but also distracting
- Ensure a single selected icon is clearly made visible when surrounded by unselected icons
- Make all icons distinctive from each other
- Ensure the harmoniousness of icons as a member of a family of icons
- Design the movement animation when dragging an icon.
- Add detailed information to an icon if possible to represent details (for instance like an empty directory having a lighter colour then a non-empty directory)
- Explore the use of combinations of icons to create new objects or actions.
- Use context sensitive help to explain functionality of icons (from Ch 13).

Direct manipulation is still evolving and new variants are created.

A first variant is that of a 3D interface. Some designers want to approach the richness of the 3D world by emulating this as exactly as possible in a 3D interface. They believe that, the close the interface resembles the real world, the easier usage will be. However, user studies show that this is not always the case and it might lead to slowdowns because of increasing disorientating navigation, too complex user actions and annoying occlusions (An example of such a problematic 3D application is the usage of 3D desktops). For a number of applications like medical, architectural, and product design this is however still a strong option.
3D interface become a lot more powerful when they do not represent the actual world, but enhance it with features that are not normally possible like x-ray vision, teleportation, multiple views of objects etc. A lot of successful games are designed this way. Also in the above

mentioned applications, features like going back and forward in time, attaching labels to objects seem at least as important as a good representation of reality.

A next variant of direct manipulation is tele-operation. This means that a physical process is being controlled by a human operator at a distance (although the computer may carry out some standard control tasks without interference). Designers must cope with problems like slower responses, incomplete feedback, increased likelihood of breakdowns etc.

Another variant is that of virtual reality. In virtual reality, part of the real world (or an imaginary world) is represented as a virtual 3D world. In a number of cases, the display is not a simple 2d plane, but a head mounted display or a room with one or more projectors, simulating a 3d environment in a more realistic way. Mostly, some form of 3d actuator like data-gloves, movement detectors etc are being used to adjust the environment to the movement of the user. This is called an immersive environment. This can allow people to be trained accurately for a number of difficult circumstances (e.g. a flight-simulator). It can also allow people to be virtually present in one room while actually be on different continents. Drawbacks are that it is still quite expensive, might require a large hardware setup. This can be avoided by using a 3D representation on 2D monitor, but it is less immersive and has the disadvantages of the above mentioned 3D interfaces.

Augmented reality is the last variant. Augmented reality enables the users to see the real world with an overlay of additional information. For instance, it might allow a user to see wires and plumbing while looking at a wall through semi-transparent eyeglasses. It might let a surgeon look at a patient while at the same time looking at a transparent overlay of an x-ray to help locate a tumour.

Chapter 7 gives an overview of somewhat more traditional forms of interaction using menu selection, form fill-in and dialog boxes.

Menus can have different organisations. You can have a singe menu, offering a choice between two or more items, or you can have a linear sequence of menu's , a tree structure of menu's, an acyclic network of menu's or even a cyclic network of menu's.

Menu's can have the form of a pull down menu, a pop-up menu or a toolbar menu. Especially when using pull-down menus representing all functions, it is important to have keyboard shortcuts available for the most used functions to enable experienced users fast access to those functions.

When menu's become large, it is possible to extend their visibility by using a fish-eye technique, that makes the few items near the cursor appear large and the rest smaller as that have a larger distance to the cursor. It is also possible to use a 2d organisation (mostly using icons) and combine that with tabs. Choices in single menu's can be shown by push buttons, radio buttons or check boxes.

A number of guidelines for creating menus:
- Use task semantics to organize menus (single, tree, cyclic etc)
- Prefer broad-shallow to narrow-deep
- Show position by graphics, numbers or titles
- Use items as titles for sub trees
- Group items meaningfully
- Sequence items meaningfully

- Use brief items, begin with the keyword
- Use consistent grammar, layout and terminology
- Allow type-ahead, jump ahead or other shortcuts
- Enable jumps to previous and main menu
- Consider online help, screen size etc.
- Provide keyboard shortcuts for frequently used items

A number of guideline for creating tree-menus:
- Create groups of logically similar items
- Form groups that cover all possibilities
- Make sure items are non-overlapping
- Use familiar terminology

The best ways to order menu's are by category or alphabetical.

Form fill-ins are uses to perform data entry from users. A number of tips to design them are:
- Give each form a meaningful title.
- Provide comprehensible instructions where needed.
- Group and sequence fields in a logical way
- Use a visually appealing and organising layout
- Use consistent terminology and abbreviations.
- Show visible space and boundaries for data entry fields
- Provide a simple and visible mechanism to move the cursor between fields using the keyboard
- Provide error correction for the user (e.g. backspace)
- Prevent errors (e.g. no alphanumerical characters in a number field)
- Give error messages for unacceptable values indicating allowable values.
- Give immediate feedback about errors where possible
- Clearly mark optional fields
- Provide explanatory messages for fields, where possible
- Let users give a clear completion signal (e.g. an OK button).

Dialog boxes are frequently used in modern GUI's to interrupt tasks of users to request users to select options or to perform limited data entry. A number of tips to design those are:
- Create groups of logically similar items
- Give each form a meaningful title.
- Group and sequence fields in a logical way
- Use a visually appealing and organising layout
- Use consistent terminology and abbreviations.
- Show visible space and boundaries for data entry fields
- Provide error correction for the user (e.g. backspace)
- Provide standard buttons (OK, Cancel)
- Smooth appearance and disappearance
- Distinguishable but small boundary
- Size small enough to reduce overlap problems; no overlap with required items
- Display close to related items
- Easy to let it disappear
- Clear how to complete or cancel

For menu's that require less pointer movement and / or smaller displays, some new types are presented. Two examples are Pie menus, setting main selections in a circle while sub selections are given by the distance from the centre, and the flow menu, also using a circle of selections, but using a return to the centre of the circle to indicate the selection of the next level in the menu hierarchy.

Chapter 8 gives an overview of the usage of command languages and of natural language.

Chapter 9 gives an overview of different interaction devices. These include: keyboard and keypads, pointing devices, speech and auditory devices, displays and printers.

The keyboard is still one of the most used, although for small devices technologies like graffiti might be as good.

Pointing devices can be divided in direct and indirect control devices. Direct control devices (e..g light pen, touch screen) are very easy to learn, but may obscure the display while using them. Indirect control devices (e.g. mouse, trackball) take more time to learn, but do not have that disadvantage.

There are some novel, mostly special purpose devices. Examples of there are eye tracking (mostly a research tool, not reliable to control a UI unless combined with manual input), multiple degrees of freedom devices (low precision, slow responses), data-gloves (heavy, confining), Ubiquitous computing and Tangible user interfaces use e.g. video cameras and (body) sensors to track and sense user movements and let systems respond to them.

The concept of speech recognition has been around for ages, but its progress is slowly and there are still many technical difficulties preventing reliable performance, especially in more or less noisy environments. The problem with speech recognition is not only technical. Voice commanding is more demanding on the users working memory that hand/eye coordination is. Also, speech requires the use of limited brain resources while hand/eye coordination is processed in another part of the brain. This can, for instance, in parallel do problem solving and planning while doing hand/eye coordination tasks. So, using a speech-interface may be more disruptive to users while carrying out tasks. Speech generation is technically reliable, but also has practical problems. Speech output has a much slower pace compared to visual output, Speech is ephemeral in nature (you can't 'glimpse back' to what just have been said) and it has difficulties providing an interface to scan/search a lot of data.

Audio interfaces to reproduce and record sound are a standard part of almost every workstation by now. Audio can be used to give feedback. This can be done by e.g. generating beeps, but nowadays this is done mostly by reproducing high quality familiar sounds during certain HCI-events. These can be viewed as auditory icons.

Displays are widely used. Earlier there were only CRT's in the range, but nowadays they are more and more replaced by LCD's. A Common resolution for LCD's these days is 1024 x 1280. Some other technological alternatives are Plasma displays, LED-displays, electronic ink and Braille displays, but they are a lot less common and mostly still in development and or expensive. All these displays are relatively small.

Large displays have been increasingly used. Especially since beamer technology has become affordable. They are not only used in command centres, but also during meetings and during education. When those displays are used interactively, normal indirect control devices like the

mouse are impractical and are replaced by, for instance, a touch sensitive screen. Also interactive elements like pull-down menu's need to be redesigned (e.g. by introducing a flow menu) since menus on a large touch screen must be compact. An example of such a large display is the SMART board that provides a large touch-sensitive screen on which a computer image is projected. Fingers can be used as a pointing device, but it also provides digital coloured pens and a digital eraser to simulate a traditional whiteboard.

Printers have been around for a long time. A lot of different printers exist, but nowadays the most used types are inkjet printers and laser printers. Some exotic printers include: photo-printers, plotters, Braille embossers and three dimensional printers.

Chapter 10 is about collaboration.

Goals of collaboration can be:
- Focused partnerships
- Lecture or demo
- Conference
- Structured work process
- Meeting and decision support
- Electronic commerce
- Teledemocracy
- Online communities
- Collaboraties
- Telepresence

There are 4 types of communication.

	Same time	**Different time**
Same place	Synchronous local, face to face (meeting rooms)	Asynchronous local (e.g. team scheduling, group calendars)
Different place	Synchronous distributed (e.g. chat, instant messaging, video / audio conference)	Asynchronous distributed (e.g. e-mail, newsgroups, listservers, discussion boards, blogs, wiki's)

Numerous of examples are given of each type of application and some evaluations are offered.

As for the same time, different place case, a reference was made to research by Microsoft on electronic classroom (2000). Four cases were compared, using text-cat, audio-conferencing, video-conferencing and face-to-face meeting. Of these, text-chat scored poorly, while both conferencing methods were quite successful and audio conferencing giving the highest satisfaction score. Face-to-face meetings discussions were considerably longer than mediated discussions. Another project, the Georgia Tech Eclass Project, emphasized the capturing of videos of lecturers so students could review them or make up for missed classes.

Chapter 11 treats the quality of service of user interfaces. It elaborates mostly on number of aspects of system response time in combination with the user expectations of these.

Chapter 12 is about the balance between function and fashion. It gives some guidelines for the use and construction of error messages, for formulating onscreen instructions, for making screen layouts and windows designs, for the use of graphical representations and for the use of colour,

Chapter 13 gives guidance for making user manuals, online help and tutorials

Chapter 14 treats the user interface aspects of information search, retrieval and visualisation.

Finally, there is an appendix about the impact of user interfaces (present and future ones) on individual and on society as a whole.

Preece, Rogers, Sharp, Interaction Design - beyond human-computer interaction, 2002 [16]

This book was less advised by a lecturer from HMI to be taken as a lead on interaction design, so only the most applicable chapter has been selected for the literature study. This was chapter 4 of this book which is about designing for collaboration and communication.

The first part explains about social mechanisms in communication and collaboration. Social mechanisms can be subdivided into thee main categories: conversational, coordination and awareness mechanisms.

Conversational mechanisms are social mechanisms to facilitate the flow of talk. They enable us to coordinate a conversation, to detect and repair breakdown and to determine differences between formal and informal conversations. Most of these mechanisms work with voice communication.

Next it combines this with a number of examples of using technology to communicate like telephone, videoconferencing and SMS. All modern collaborative technologies are summarized as Computer Mediated Communication (CMC).

This is devisable in three subtypes of CMC:

- Synchronous communication (e.g. video conferencing, instant text messaging, Collaborative Virtual Environment). Benefits: relatively cheap, fast, may increase people's confidence. Drawbacks: lack of bandwidth, difficult to establish eye contact, abuse is possible when people can hide behind an alias or avatar.
- Asynchronous communication (e.g. email). Benefits: Ubiquity, Flexibility, powerful.
- CMC combined with other activities (e.g. customized meeting rooms, networked classrooms, shared authoring and drawing tools). Benefits: supports multi-tasking since other activities besides talking are supported, increases efficiency, increases awareness. Drawbacks: Difficulty determining what other persons are looking at or working on, floor control must be more explicit to avoid conflicts.

Coordination mechanisms are important to keep meetings organised during collaboration. These can be subdivided in 3 categories:

- Verbal (needs audio) and non verbal (needs video).
- Schedules, rules and conventions.
- shared external representations (e.g. checklists, shared calendars, document sharing)

Awareness mechanisms are social mechanisms to allow people in a group to find out what is happening in that group. This means to find out what others are doing, who is talking to whom and to let others know what is happening. These mechanisms can be very well supported by a lightweight chat application.

Problems might arise when people violate conventions of an online system out of "laziness", Also people tend to discard coordination mechanisms when they do not find them socially acceptable. This means the right balance has to be found using system coordination by not giving the users too much freedom, but also by not forcing the users against their will.

R Rienks, Meetings in Smart Environments, draft 2007 [17]

This thesis, as stated in the first chapter, finds its origin in the AMI (Augmented Multi-party Interaction) project that aims the development of technology that facilitates multi-modal interactions in the meetings domain. Within the AMI project, research is done on e.g. human-human communication modelling, speech recognition, computer vision, multimedia indexing and retrieval. Its main aims are to develop technologies for the disclosure of meeting content and the provision of live meeting support.

One of the major deliverables of the AMI project is a meeting corpus consisting of a large number of recorded and annotated 4 person face-to-face meetings that can be used to do research on.

This thesis itself aims at the assessment of two aspects of high level meeting phenomena: Influence hierarchies and argument structures in a four person face to face meeting (using the AMI meeting corpus).

Chapter 2 contains a general introduction to meetings

In chapter 3, this thesis gives an interesting oversight of different types of meetings.

One way to classify meetings is the difference of participants in time and space. Standard teleconference meetings mostly address the difference in space by transferring video, audio and information directly, but a difference in time is also possible when not every member of a group that has a meeting can attend at the same time. Another classification is in the usage of computer produced stimuli ranging from reality, augmented reality, augmented virtuality to a virtual environment.

Furthermore, this chapter gives an oversight of a number of technologies to assist meetings. Two major technologies for meeting support during the meeting itself are Group Support Systems and Software Agents. Next to that, pre and post meeting support is an important topic. Technologies to support this are scheduling systems to support appointments for the meeting itself and appointments made during the meeting for follow-ups. Also various browsing technologies to review the meeting (audio, video and other information) are important. These can be enhanced by technologies that support automatic selection of important moments in a meeting or even summarize all important information of a meeting in a document. These functions might be used to review the meeting later or for participants to be able to join a meeting later and catch up. Furthermore, these technologies can be enhanced by automated analyses of the meeting in terms of the interactions between participant to increase understanding of how what was going on and how decisions were arrived at.

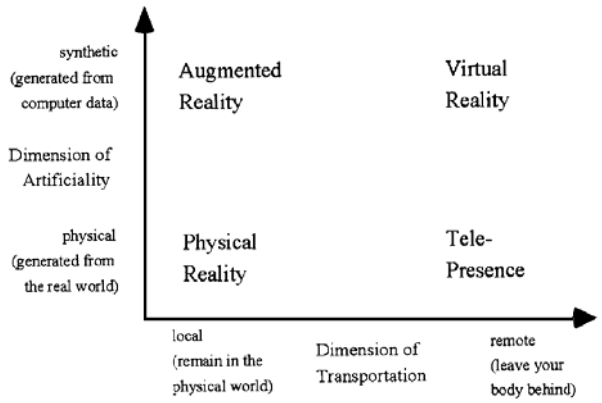
Finally some aspects of Computer Mediated Communication are treated. From earlier research it appears that people are flexible and adapt their communication style to the availability of different media (an example is the use of emoticons in electronic text-only communication). Differences arise during group interaction. For instance, during video mediated meetings, people

adapt a more formal mechanism of turn-taking than during a real face-to-face meeting. Finally, it seems that actual presence during a meeting is largely influenced by the richness of the available media. Personal cues are better transferred using video signals that using text or audio-only channels. This poses problems for meetings that are more moving towards a virtual environment since it is difficult at present to detect and convey all social cues into such an environment.

Chapter 4 deals with general feature extraction and classification done on the AMI corpus. Several machine learning algorithms are exploited like Support Vector Machines. Chapter 5 deals with classification to reveal influence hierarchies by assigning and order to the participants from the lowest to the highest influential person. Chapter 6 deals with classification to reveal argument structures and how to represent and interpret them. For both phenomena it is clear that more research is needed to be able to have a fully automated and reliable feature extraction and classification. Chapter 7 tries to relate the two phenomena to each other and, although there seems to be some correlation, it is not enough to use this to enhance machine learning algorithms to increase the quality of classification for each phenomenon.

Chapter 8 gives a view on future meeting technology where computers, both in real and virtual meetings more and more assist the users. This may be in the form of automatic adaptations of the environment (real or virtual), the availability of extra real time information and analyses of the meeting so far, or in the form of autonomous agents, like for example a virtual chairman, interacting with both the participants and the environment.

A new way to categorize meetings is given, combining the two models that can be used to categorize meetings. Meetings were already divided in time and space and along the continuum between reality and virtuality. A categorisation that combines these two, with the focus on synchronous communication is shown below.



Meetings in augmented reality might in the future be possible by using electronic glasses to superimpose remote participants on the real environment as if the were there or by using 3D displays / holograms. Tele presence means the users act on a real, sensed environment in both places. This might for instance be done by using a robot, acting on sensed movements at the remote site. Virtual reality means the environment and the projection of the users in that environment are artificially constructed. The real world movements of the users are, more or less, translated to those artificial versions.

ISO/IEC JTC1/SC29/WG11, MPEG-4 Overview - (V.21 – Jeju Version), 2002 [18]

MPEG-4 is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group) and provides the standardized technological elements enabling the integration of the production, distribution and content access for Digital television, interactive graphics applications (synthetic content) and interactive multimedia (World Wide Web, distribution of and access to content).

MPEG-4 audiovisual scenes are composed of several media objects, organized in a hierarchical fashion. At the leaves of the hierarchy, we find primitive media objects, such as still images (e.g. as a fixed background), video objects (e.g. a talking person - without the background and audio objects (e.g. the voice associated with that person, background music). A number of such primitive media objects are capable of representing both natural and synthetic content types, which can be either 2- or 3-dimensional. In addition to the media objects mentioned above, it defines the coded representation of objects such as text and graphics, talking synthetic heads and associated text used to synthesize the speech and animate the head; animated bodies to go with the faces and synthetic sound. Furthermore it provides a standardized way to describe a scene, allowing for example to place media objects anywhere in a given coordinate system, apply transforms to change the geometrical or acoustical appearance of a media object; group primitive media objects in order to form compound media objects, apply streamed data to media objects, in order to modify their attributes and change, interactively, the user's viewing and listening points anywhere in the scene.

To achieve synchronized delivery from source to destination MPEG4 specifies a synchronization layer and a delivery layer containing a two-layer multiplexer. These provide mechanisms to timestamp and multiplex different elementary streams and mechanisms to exploit different QoS as available from the network.

The MPEG4 standard is meant to be extendible. New versions will appear in the future, but those will always be backwards compatible with older versions. This is done by means of profiles. New versions add more profiles, but never change existing ones.

Mpeg4 can be used to model a 3d environment using VRML like definitions and to navigate in such an environment, but it can also be used to only stream audio and video with relative high quality using for instance only the advance simple profile. It then takes care of transport and multiplexing / demultiplexing of audio and video streams. It is also possible to send text and/or other objects using mpeg4 and to multiplex / synchronize this all in one stream.

Streaming video using the simple profile or the somewhat better advanced simple profile from mpeg4 gives a high quality video stream while at the same time using relatively low bandwith. Known applications using mpeg4 are, amongst others: Divx, Windows Media and Apple Quicktime.

Xu, G. Sugimoto, T. , Rits Eye: a software-based system for real-time face detection and tracking using pan-tilt-zoom controllable camera, 1998 [19]

In this report a system is described that is able to detect faces and to actively follow these using a pan-tilt-zoom camera. Face detection is mainly done by detecting skin colours and combine those with detection of dark regions for eyebrows, eyes and the mouth at certain positions. It can be made robust against having different racial skin colours, being of different sex and wearing different hairstyles or glasses. A face does not need to look exactly straight at the camera, but enough part of the face need to be visible.

Ronald Poppe, Rutger Rienks, Parlevison: Rapid Development of computer vision Application draft, website, 2006? [20]

Parlevision is a software utility for the rapid prototyping of computer vision applications. It uses a graphical user interface to utilize Intel's Open-CV image processing library and enhances this with many higher order functions. The Parlevision system is used in many applications including tracking of hands, estimating human poses, corridor surveillance, analysing beats and analysis of facial expressions.

Chris Aniszczyk, Eclipse DeveloperWorks Article, 2006 [21]

Eclipse is an open source community whose projects are focused on providing an extensible development platform and application frameworks for building software using Java. Amongst others, Eclipse provides a platform to quickly prototype, collaborate, and share ideas built on a common architecture.

Users of Eclipse-based offerings benefit from:

- Having access to research and knowledge from the entire Eclipse ecosystem
- The ability to reuse skills because of the consistent Eclipse interface

Java technology developers using Eclipse benefit from:

- A good and extensive Java IDE
- Native look and feel across platforms
- Easy extensions to Java tooling

Developers of Eclipse tools benefit from:

- A portable and customizable platform
- Seamless tool integration
- An end-to-end solution

This article gives links to the latest version of Eclipse, information on IBM's involvement with Eclipse, and a guide to some of the most interesting Eclipse projects.

Jesse James Garrett, Ajax: A New Approach to Web Applications, 2005 [22]

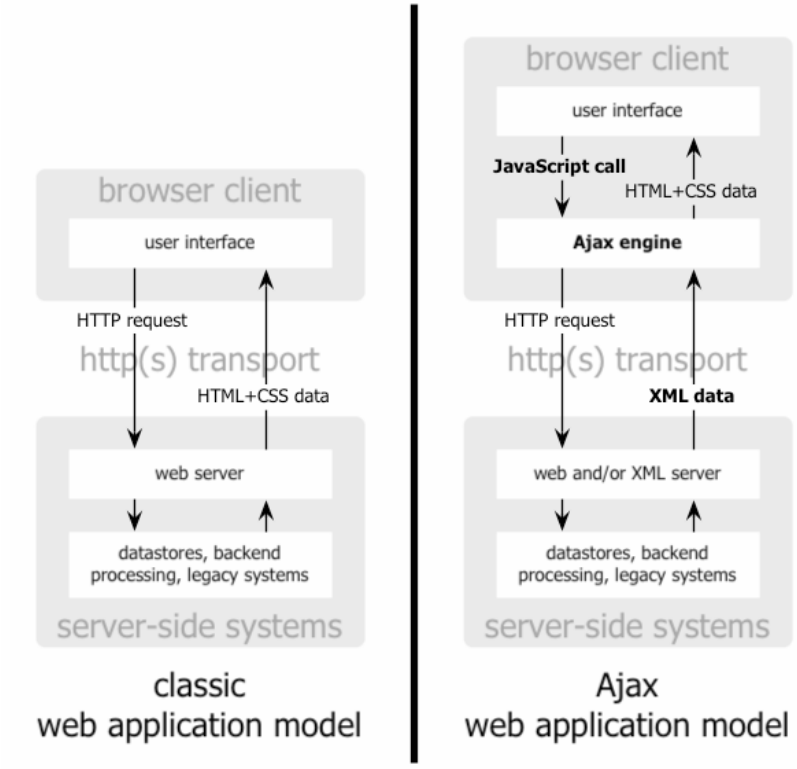
This is an introductorily article about AJAX. It explains how it is composed, what the differences are between AJAX and classical web applications, and gives some examples of sites that use AJAX.

Ajax isn't a technology. AJAX is an acronym that stands for Asynchronous JavaScript And XML. It's really several technologies, coming together in powerful new ways. Ajax incorporates:

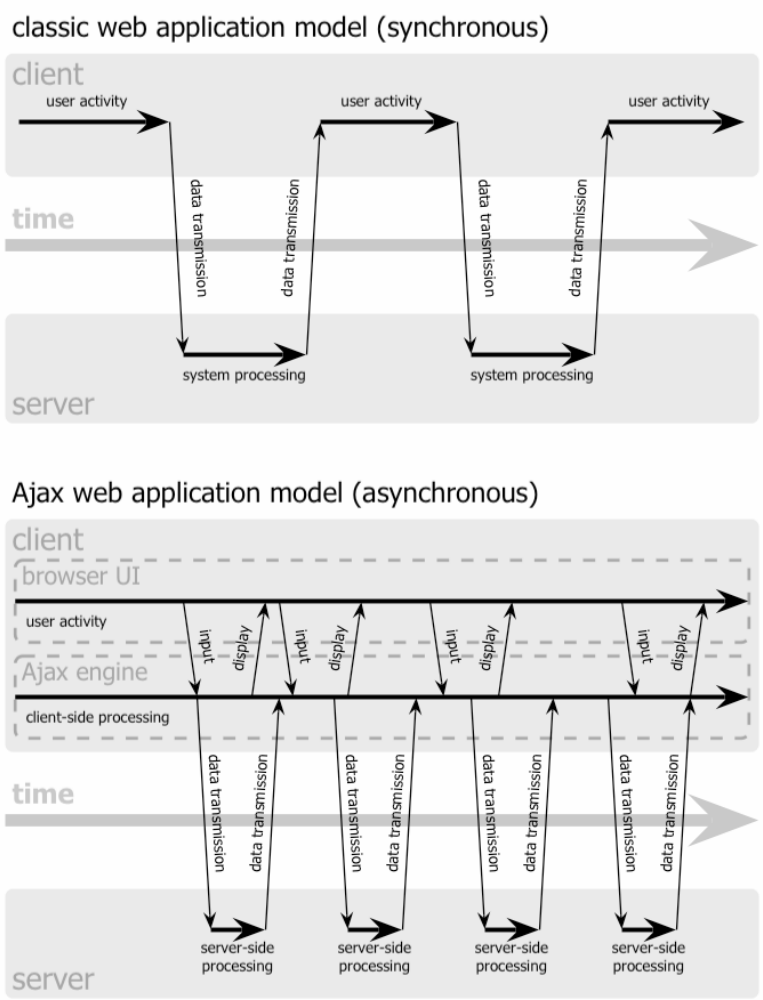
- Standards-based presentation using XHTML and CSS.
- Dynamic display and interaction using the Document Object Model.
- Data interchange and manipulation using XML and XSLT.
- Asynchronous data retrieval using XMLHttpRequest.
- JavaScript is binding everything together.

An AJAX application eliminates the start-stop-start-stop nature of interaction of classical web applications, due to having to reload the whole webpage, by introducing an intermediary, an AJAX engine, between the user and the server.

Instead of loading a webpage, at the start of the session, the browser loads an AJAX engine, written in JavaScript. This engine is responsible for both rendering the interface the user sees and communicating with the server on the user's behalf. The Ajax engine allows the user's interaction with the application to happen asynchronously, i.e. independent of communication with the server.



The traditional model for web applications (left) compared to the Ajax model (right).



The synchronous interaction pattern of a traditional web application (top) compared with the asynchronous pattern of an Ajax application (bottom).

Some examples of websites already using AJAX technology are: Gmail, Google Maps, and Flickr.

Paulson, L.D., Building rich web applications with Ajax, 2005 [23]

This article explains the basics of AJAX like in [22], but goes into more detail about the different components used by AJAX.

Dynamic Hypertext Markup Language (DHTML): The technology describes HTML extensions that designers can use to develop dynamic Web pages that are more animated than those using previous HTML versions. For example, when a cursor passes over a DHTML page, a colour might change or text might get bigger. Also, a user could drag and drop images to different places.

Extensible Markup Language (XML): Ajax uses XML to encode data for transfer between a server and a browser or client application. XML is a markup Meta language that can define a set of languages for use with structured data in online documents.

Cascading StyleSheets (CSS): CSS gives Web site developers and users more control over how browsers display pages. Developers use CSS to create stylesheets that define how different page elements, such as headers and links, appear. Multiple stylesheets can apply to the same Web page.

Document object model (DOM): The DOM is a programming interface that lets developers create and modify HTML and XML documents as sets of program objects, which makes it easier to design Web pages that users can manipulate. The DOM defines the attributes associated with each object, as well as the ways in which users can interact with objects. DHTML works with the DOM to dynamically change the appearance of Web pages. Working with the DOM makes Ajax applications particularly responsive for users.

JavaScript: Ajax uses asynchronous JavaScript, which an HTML page can use to make calls asynchronously to the server from which it was loaded to fetch XML documents. This capability lets an application make a server call, retrieve new data, and simultaneously update the Web page without having to reload all the contents, all while the user continues interacting with the program. Because JavaScript is a cross-platform scripting language, Ajax applications require no extra plug-ins.

XMLHttpRequest: Systems can use JavaScript-based XMLHttpRequest objects to make HTTP requests and receive responses quickly and in the background, without the user experiencing any visual interruptions. Thus, Web pages can get new information from servers instantly without having to completely reload

This article also compares AJAX with some other available technologies:

Microsoft: Microsoft is reportedly trying to simplify the development of rich Web applications via a project code-named Atlas. Atlas will provide tools to be used with the company's ASP.NET, which developers use to create Web pages whose elements are treated as objects.

Flash: Macromedia's Flash is a popular type of Web authoring software that creates vector-graphics-based animation programs. Unlike Ajax, Flash supports audio and video, but users need the (former Macromedia, now Adobe) Flash plug-in to be installed.

Java: Java-based applications offer some advantages over Ajax-based programs. For example, there are reusable Java components and toolkits, which is not yet the case for Ajax. Java might be used on the server and JavaScript in the browser."

Furthermore, it gives some advantages and disadvantages of AJAX technology. One advantage is that AJAX web applications perform better than standard web applications because they are more responsive towards users.. Furthermore, they make more efficient usage of available bandwidth. Another advantage of AJAX is that many designers already have experience with a number of its components and a big advantage of AJAX is that it is platform independent. A drawback of AJAX is that, once all its techniques are combined in a complex project, it can become an extremely complex system that needs very experienced designers. Another disadvantage is that the toolkits and frameworks for AJAX are not mature. Another drawback is that AJAX raises security concerns because its component technologies are now used in a different way that might open security holes. A last drawback is that JavaScript in some cases can be browser dependent, forcing programmers to take differences between browsers into account.

Cynthia Calongne, Jeff Hiles, Blended Realities: A Virtual Tour of Education in Second Life, 2007 [24]

This article first describes very briefly the Second Life concept, the possibilities for support for educators and then continues with a number of examples of educational use of Second Life.

Second life is a 3D virtual world founded by Linden Lab of Linden Research that has a steeply rising user base that now reaches over 2 million. The Second Life software provides free accounts with the ability to design, integrate and texture structures, furnishings, clothing, and avatars using in world resources. Sandboxes offer open spaces for creativity, collecting more free items and sharing ideas. The resources available to SL users include whiteboard and communication tools, language translators, information centres, and support from a variety of educational groups. A wide range of free software tools are available,

In Second Life people use an avatar to represent themselves. The can define how their avatar looks like and they can create all kinds of virtual 3 dimensional objects using primitives like cubes or cylinders. The 3D environment is organized in the form separate simulated environments, called islands.

Support for education can be found in a number of locations outside of Second Life in the form of wiki's, blogs, mailing lists, or within Second Life as libraries on e.g. the Information Island.

Examples of educational use are:

- Teaching architecture by letting students design 3D buildings and objects, sometimes even combining this with art and music, creating multidisciplinary assignments.
- Letting students using Second Life to gather inspiration and experiences to write stories for English literature
- Combining Second Life with Skype, blogs and Blackboard to give lectures
- Simulating a courtroom letting the students play the roles of judges, lawyers etc.
- Letting students use the powerful and flexible scripting, texture and animation options of Second Life to explore system engineering methods, prototyping and improve their problem solving skills.

Cheng-chao Su, An open source platform for educators, 2005 [25]

This article focuses on setting up an online educational platform to assist both teachers and students using open source software. The author has experimented with a number of systems and selected those that were most easy to install and manage. The reason the author stresses the use of open source software because of a number of reasons: These systems are free, users have full control and maximum customization, open source systems are extendible, there is mostly a friendly online supporting group present and there is a large range of educational open source software available.

It starts with the basics: a platform using an Apache server with PHP and SQL that can be running both on a Linux or a Windows operating system.

Next a number of services are added: a WIKI (MediaWiki) to collaborate, a blog (Wordpress) to make web logs, a bulletin board system (YaBB, phpBB or Simple Machines) to maintain a forum, a content management system (XOOPS) or a content and course management system (Moodle) to create dynamic websites and to manage courses and a Multiple User Dimension (MOO) to allow users to construct and manipulate objects and interact with both other users and objects.

The article next shows the reader a number of existing online learning communities that can be used to find information and / or gain experience about setting up, maintaining and using such tools.

At the end the author stresses that usage of free open source learning platforms can be a means to narrow the existing information gap for developing countries.

S. Graf, B. List, An evaluation of open source e-learning platforms stressing adaptation issues, 2005 [26]

In this article 36 open source e-learning platforms were compared. Of those 36, after a pre evaluation phase, 9 were selected for the final comparison. In this comparison, a lot of general features were compared, but also a specific set of features for adaptation capabilities of the platforms. These features contain criteria like adaptability and extendibility. Of those 9 platforms, Moodle scored highest for both the general features and for adaptation capabilities. Especially for extendibility Moodle scored very high.

Peter Wayner, Top AJAX tools deliver rich GUI goodness, 2006 [27]

This article compares a number of commercially available AJAX toolkits: Backbase, Bindows, JackBe NQ Suite, and Tibco General Interface.

All offer broad widget collections, rich tools, sophisticated debuggers, and development platforms. All of these systems are built around full collections of widgets that are joined with a central backbone of events and server calls that link the widgets in a cohesive set of panes. The major differences among these packages lie not in the capabilities, but in the server support and the finer details of their approach.

Of these, Tibco General Interface 3.2 does not holds the best overall scores, but it comes close enough and the fact that it is the only commercial package listed here that is available for free under a BSD open source licence makes it the only viable choice for usages of the listed packages. Another advantage of Tibco General is the fact that it is independent from server code makes it easier to integrate with other systems.

This article also compares them with their open source counterparts of which a number were compared in another article [34] of the same author.

IBM HiPODS Latin America Team, Web 2.0 -- Advanced AJAX Applications with the Google Web Toolkit, 2007 [29]

This document shows an example of the use of the Google Web Toolkit (GWT) to build an AJAX application. The example project consists of a contact list composed of a paged table and a photo area. It can show a dialog box with extra information on a contact selected from that table.

This project is made using Eclipse, IBM Rational developer for Websphere as an Integrate Development Environment (IDE), and IBM Cloudscape for database access. The source code for this example project is downloadable for study.

The explanation is quite detailed in how, for instance, to customize a dialog box and contains lots of example code showing how to implement details of the GUI.

Next it shows how GWT makes AJAX-calls transparent for the programmer by using the GWT remote services infrastructure to place Remote Procedure Calls (RPC) that can be used to exchange Java-objects between the client |(browser) and java code on the server.

This article shows in general how GWT works and can be used, however it focuses very much on the already created example and does not fully explain how to use GWT to convert java into JavaScript, or how to stat building a GWT application from zero. In order to actually start using GWT, some more literature would be needed.

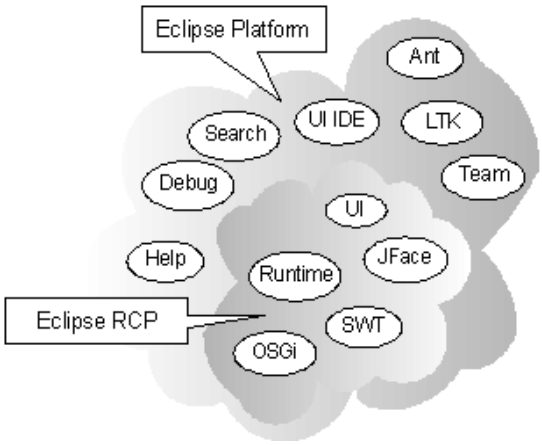
IBM, Eclipse Platform Technical Overview, 2006 [30]

The Eclipse Platform is designed for building integrated development environments (IDE's), and arbitrary tools. This paper is a general technical introduction to the Eclipse Platform.

Overview

In its entirety, the Eclipse Platform contains the functionality required to build an IDE. However, the Eclipse Platform is itself a composition of components; by using a subset of these components, it is possible to build arbitrary applications. One of the key benefits of the Eclipse Platform is realized by its use as an integration point. Building a tool or application on top of Eclipse Platform enables the tool or application to integrate with other tools and applications also written using the Eclipse Platform. The Eclipse Platform is turned in a Java IDE by adding Java development components (e.g. the JDT) and it is turned into a C/C++ IDE by adding C/C++ development components.

The Eclipse Rich Client Platform (RCP) is one of the above mentioned subsets of components.



The Eclipse Rich Client Platform (RCP).

As the name "rich client" implies, Eclipse RCP is an excellent platform for building applications that work in conjunction with application servers, databases, and other backend resources to deliver a rich user experience on the desktop.

Although the Eclipse Platform has a lot of built-in functionality, most of that functionality is very generic. It takes additional tools to extend the Platform to work with new content types, to do new things with existing content types, and to focus the generic functionality on something specific.

The Eclipse Platform is designed and built to meet the following requirements:

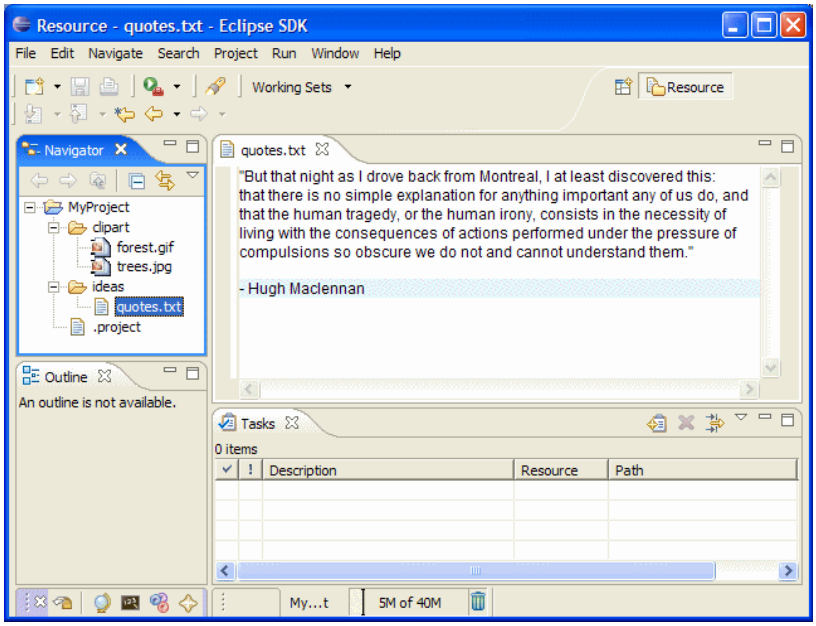
- Support the construction of a variety of tools for application development.
- Support an unrestricted set of tool providers, including independent software vendors (ISVs).
- Support tools to manipulate arbitrary content types (e.g., HTML, Java, C, JSP, EJB, XML, and GIF).
- Facilitate seamless integration of tools within and across different content types and tool providers.
- Support both GUI and non-GUI-based application development environments.

- Run on a wide range of operating systems, including Windows®, Linux, Mac OS X, Solaris AIX, and HP-UX.
- Capitalize on the popularity of the Java programming language for writing tools.

The Eclipse Platform's principal role is to provide tool providers with mechanisms to use, and rules to follow, that lead to seamlessly-integrated tools. These mechanisms are exposed via well-defined API interfaces, classes, and methods. The Platform also provides useful building blocks and frameworks that facilitate developing new tools.

Basic workings and architecture

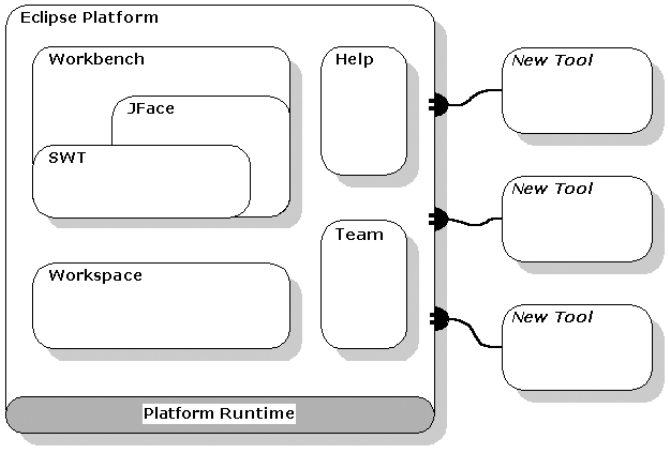
The most obvious thing that the Eclipse Platform provides is a managed windowing system. User interface components are part of this (including entry fields, push buttons, tables, and tree views). The platform also provides window lifecycle management, docking views and editors, the ability to contribute menu items and tool bars, and drag and drop.



The Eclipse Platform User Interface

The navigator view (top left) shows the files in the user's workspace; the text editor (top right) shows the content of a file; the tasks view (bottom right) shows a list of to-dos; the outline view (bottom left) shows a content outline of the file being edited (not available for plain text files).

Eclipse consists of a number of standard modules that can be extended by an arbitrary number of plug-ins. The Basic Eclipse architecture is given below:



Basic Eclipse Platform architecture

Workspace: The various tools plugged in to the Eclipse Platform operate on regular files in the user's workspace. The workspace consists of one or more top-level projects, where each project maps to a corresponding user-specified directory in the file system.

SWT: The Standard Widget Toolkit (SWT) provides a common OS-independent API for widgets and graphics implemented in a way that allows tight integration with the underlying native window system. The entire Eclipse Platform UI, and the tools that plug in to it, use SWT for presenting information to the user. SWT addresses the tension between portable toolkits and native window system integration by defining a common API that is available across a number of supported window systems. For each different native window system, SWT uses native widgets wherever possible; where no native widget is available, SWT provides a suitable emulation. Common low-level widgets such as lists, text fields, and buttons are implemented natively everywhere. But some generally useful higher-level widgets may need to be emulated on some window systems.

JFace: This is a UI toolkit with classes for handling many common UI programming tasks. JFace is system-independent, and is designed to work with SWT without hiding it. JFace includes the usual UI toolkit components of image and font registries, dialog, preference, and wizard frameworks, and progress reporting for long running operations. Two of its more interesting features are actions and viewers. The action mechanism allows user commands to be defined independently from their exact whereabouts in the UI. An action represents a command that can be triggered by the user via a button, menu item, or item in a tool bar. Each action knows its own key UI properties (label, icon, tool tip, etc.) which are used to construct appropriate widgets for presenting the action. This separation allows the same action to be used in several places in the UI. Viewers are model-based adapters for certain SWT widgets. Viewers handle common behaviour and provide higher-level semantics than available from the SWT widgets.

Workbench: This provides the UI personality of the Eclipse Platform, and supplies the structures in which tools interact with the user.

Help: Eclipse has a platform help mechanism to use and define online documentation.

Team: The Eclipse Platform allows a project in the workspace to be placed under version and configuration management with an associated team repository.

Gary Horen, Exploring Ajax Runtime Offerings, 2006 [31]

This article begins by describing AJAX, but that has already been treated in several other previous articles. It further state that development using raw Ajax is difficult because of inter-browser variability, and also because of the relative immaturity and the browser-dependent state of the art of the available development tools that work at that level. It therefore recommends the use of one of the available AJAX runtime frameworks.

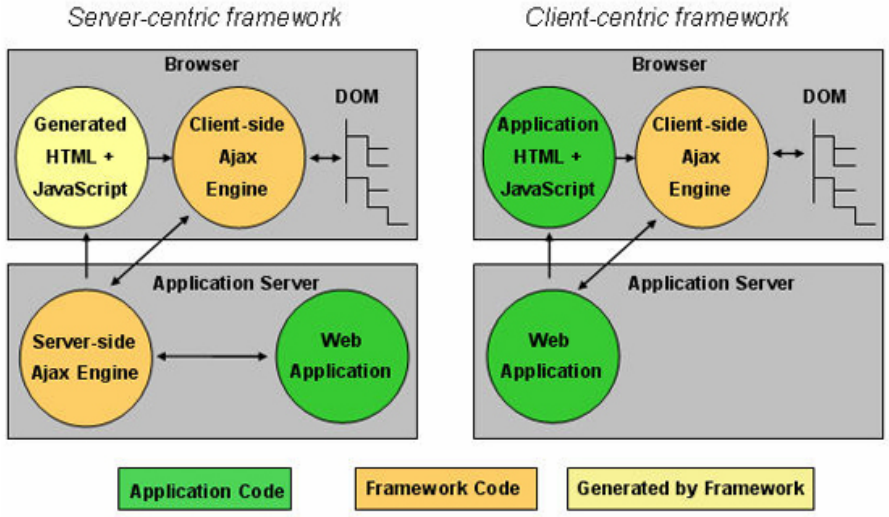
This article tries to classify those frameworks in a number of ways:

First it compares open source to commercial software. Commercial frameworks are more mature in many cases, mostly they have better documentation available and vendors often have a stronger commitment to backwards-compatibility. Open source frameworks tend to be somewhat more uneven and are driven mostly by the focus of the developer's community. However, open source can be more flexible since it is possible to add new functionality yourself.

Second, comprehensive frameworks are compared to sets of individual components. Comprehensive frameworks work very well when one builds a new application while extending existing ones might sometimes be easier using lose components. Frameworks perform best if they have total control over a webpage while components might be easier to integrate with other frameworks and/or portals. Lose components might require more programming since they usually operate at a lower level.

Next, a distinction is made between declarative and procedural frameworks. Declarative frameworks provide means for a developer to describe the layout of a GUI and the relations between its components so the framework knows how to handle events from those descriptions. Procedural frameworks however, doe only provide API's that enable a developer to program code that handles GUI-events.

A quite important distinction between frameworks is in those that are client-centric and those that are server-centric.



Server-centric vs. client-centric framework architecture

The client-centric frameworks were the first to be developed while the server-centric frameworks emerged later. In a server-centric framework, the whole application is developed as server-side

code while the framework can take care of the actual separation in a client- and a server-side, providing API's to receive and push back events between the client and the server. In a client-centric framework, the focus is almost completely on the client side. The AJAX code executes on the clients browser and communicates with the server–side components that are developed independently of the clients AJAX-framework meaning both sides have to be programmed separately. A server-centric framework often holds a representation of the browsers DOM and widget states on the server, making it easier to restore the state of AJAX sessions when users leave and return later (or when a connection breaks).

This division can be extended with an extra subdivision. The client-centric frameworks can be divided in frameworks that concern themselves mostly with the plumbing of Java-code without much concern for the GUI and in higher level frameworks that concern themselves with the GUI and include some kind of AJAX rendering engine.

The server-centric frameworks on their turn can be subdivided in frameworks that still work like a web-application with markup usage and those that do not. The frameworks that work like a web-application offer GUI layout in the form of tags and event handling in the form of declarative terms in the layout tags and in the form of Java. Mostly JSF is used as a programming model. The frameworks that do not work like a web application, offer a uniform Java API that functions and looks mostly like Swing. A developer constructs the GUI using Java objects and by handling events in Java. It does hardly matter anymore whether the final application runs in a browser using javascript or runs as a true java desktop application. This type of framework is harder to use in combination with other web-applications because it acts as a separate program and does hardly take html in account anymore.

Server-centric		Client centric	
Java	UI + Java	Client-centric Framework	JavaScript
•Java UI definition •Java event handlers	•UI Tags •Java event handlers	•Framework for page layout •JavaScript event handlers	•JavaScript event handling, plumbing, and UI layout
•Inoopract RAP •Google Web Toolkit (GWT) •Echo2	•IceSoft •Backbase JSF •MyFaces Tomahawk	•Dojo •OpenRico •JackBe	•DWR •Prototype •Sajax

Server-centric vs. client-centric framework table

Applications that are more dependent on data entry and server interaction can greatly benefit from these server-centric framework, but applications that need more graphics and very fast even processing are probably better implemented using a client-centric framework.

Frameworks that offer GUI-features can differ much in the richness of the widget set to construct the GUI from and they differ in the presence or absence of tools to construct and layout the GUI.

A few frameworks are examined closer:

Prototype is a low-level library of JavaScript functions. It is open source, client-centric, consists of individual components en is procedural of nature and has no means to specify widgets etc. On

top of Prototype, a few somewhat more enhanced open source frameworks like OpenRico and Scriptacious were build to enable widget handling. Icefaces is a commercial product build on top of Prototype and Scriptacious.

Direct Web Remoting (DWR) is open source, has a server-centric and a client-centric component, consists mostly of lose components and is somewhat procedural. It has no widget handling functions and no GUI design tools.

Dojo is open source, client-centric, can be used both as a comprehensive framework or as lose components, and is mostly procedural. It has widget handling features.

Backbase is a commercial comprehensive framework that can be client-centric or server-centric, dependent of the edition that is used. It is mostly declarative and has a rich widget set.

The Google Web Toolkit (GWT) is a comprehensive server-centric open source framework that is procedural in nature. It uses a compiler to translate Java code into JavaScript, so programming can be done in Java. It has no widget handling features of its own. These can be added by using a commercial package like GWT designer from Instantiations.

Next to a framework, in many case some extra tools are needed. In the case of client-centric frameworks, it should handle functions like a JavaScript debugger/ validator, DOM inspection, message tracing etc. One of the better tools for this purpose (although still in beta) is the Eclipse Ajax Tooling Framework (ATF). In case of server-centric frameworks, the same Java and JSP tooling, like Eclipse, can be used as for traditional applications.

Project expert op afstand, videoconferencing tips en scenario's, 2006 [32]

The project "expert op afstand" (expert at a distance) has a lot of documentation about using teleconference systems for educational purposes. It describes a number of different systems that can be used, give general advice about adaptation of the environment for effective teleconferencing and gives advice about how to handle different aspects of a teleconference session from the viewpoint of different roles (like listening as a student, maintaining control of the session as a teacher, preparations to be done for an expert that is consulted by one or more classes with teachers, etc). Furthermore, a number of scenarios are given, from the perspective of the teacher to guide different types of teleconference sessions featuring an expert.

Light, sound and positioning

A number of advices about light, sound and positioning are useful in any teleconference situation.

This is especially true for the usage of light and sound. The level of light in this case is less important then the distribution of light. For instance, to light a face during teleconferencing one should never use back lighting or light from below, but light from about 60 degrees above.

Like this  and not like  or 

Also, avoid light the participants can look directly into so it blinds them.

As for sound, it is most important to avoid echoes. This can be done by using a headset (this usually gives the best results) or by using special echo-cancellation equipment when using normal microphones. Furthermore it is strongly advisable to use a room that has only a minimal amount of background noise (by e.g. switching of fans and air-conditioning systems) and to prevent a room sound to "hollow" by e.g. finding a room with less straight corners, mounting objects to the wall, putting carpet and curtains in the room etc.

The best view for a camera is when it is positioned somewhat above eye-height.

Room-based situations and scenarios

Most of the situations presented in "experts at a distance" imply a situation that on one side, you have an expert having different roles (giving a demonstration, coaching, teaching etc) who uses one teleconference system on his own and a number of classrooms, each using one stand-alone teleconference system equipped with a pan-tilt-zoom camera for the whole room. Especially in a primary school and probably also in a high school this is probably a desirable situation using minimal network bandwidth and equipment and in which each class can be strongly socially controlled by a teacher actually present in each room. In most scenarios presented here, these teachers have a preparing and coordinating role, assisting the expert to maintain control over the meeting.

Second Life WIKI, Linux Viewers and 3D Acceleration, 2007 [33]

https://wiki.secondlife.com/wiki/SLDev-Traffic_1#Linux_Viewers_and_3D_Acceleration

Without modern, hardware based 3D acceleration, 3D applications such as second life run too slow to be useable.

Peter Wayner, Surveying open-source AJAX toolkits, 2006 [34]

This article examined six open source toolkits to create AJAX applications.

All six examined packages offered a number of extremely useful user interface widgets and background tools for simplifying the process of building an AJAX application.

They will be most interesting to developers with running applications who only want to add a new part or update a page. If you want to add an animated panel or a live table, for example, you can usually cut and paste them into place; the examples are generally good enough to help you accomplish the basic routines.

Still, there's often a need to dive into the code if you want to do things that are even slightly different from the established framework. For instance, with some of these tools it took just a few minutes to add a table of data that could be sorted on the client, if the example code was imitated as closely as humanly possible. But if you try to improvise or do something differently, the code would break, and the documentation was often quite unhelpful.

This rough surface means that any development team should think about its environment before putting open source AJAX tools into action. If you have seasoned programmers and the ability and time to use the freedom and flexibility provided by the source code, these toolkits are great deals. If you're a newer programmer or someone without the time to wade into a project, you should think twice about the costs and consider the more professional packages.

Andrey Filippov, AJAX, LAMP, and liveDVD for a Linux-based camera, 2006 [35]

This article describes a web interface for an embedded device, a network camera, that uses AJAX to control video settings and that integrates the video window in the browser window. To achieve this in Mozilla Firefox, the Genres plugin (http://savannah.nongnu.org/projects/genres/) for Mozilla browsers is used to control an external video viewer called Mplayer.

It also shows how to incorporate and control an external playing and recording system.

Microsoft, Using the Windows Media Player Control in a Web Page, 2007 [36]

This article describes how to embed the media player in a webpage using Internet Explorer and how to control the embedded player using (amongst others) Javascript.

Jacqueline Emigh, New Flash player rises in the web-video market, 2006 [37]

In this article Adobe's Flash software is presented as a means of playing and streaming video contend from within web-applications.

Flash is available for many platforms and, especially since version 8, it offers very efficient and high quality video streaming capabilities. This is achieved by using the Spark codec from Sorenson Communications and the VP6 codec from On2 Technologies. Especially the VP6 codec has many advantages like using fewer resources at the client side and being able to gradually adapt to lower connection speeds without user intervention. VP6 includes both an advanced profile for high-end, high band-with usage and a simple profile for low-end low band-with usage. Flash can seamlessly integrate with many browsers and has interaction possibilities much like AJAX. Streaming can be done using products like Flash Media Server.

An extra advantage of Flash is that it is already available on many systems without the need to install additional software. A study from the NDP group showed its installed base is very large, in fact, larger than, for instance, Quicktime.

Cristian Darie, Bogdan Brinzarea, Filip Cherecheş-Toşa, Mihai Bucica AJAX and PHP: Building Responsive Web Applications, 2006 [38]

This book describes the basics of programming AJAX interfaces using JavaScript, DOM, CSS, and XML on the client side and PHP / MYSQL on the server side.

It starts with a general introduction to AJAX and its possibilities. Next it continues with the client side techniques needed for AJAX. These include using JavaScript, DOM, the XMLHttpRequest object and XML This chapter is not a complete tutorial, but handles the foundations. Next it treats the foundations of server-side techniques using PHP and MySQL and goes deeper into techniques for implementing common tasks, handling JavaScript security and error-handling.

Furthermore, it includes a lot of examples with detailed explanation. Among those is a chat application based purely on AJAX technology.

In an appendix, it show how set up a working server environment for using with AJAX client applications

Miguel de Icaza, Jaime Anguiano, Lluis Sánchez, Niel Bornstein, Mono FAQ: Technical, Mono-project, 2007 [39]

Is Mono Binary Compatible with Windows?

Yes, Mono is binary compatible with Windows, which means that you can run binaries produced by .NET compilers from Microsoft and other vendors.

When porting your applications, you should make sure that you test its functionality as differences in the underlying operating system and differences in the VM implementations (bugs, missing features) might affect your application.

Mono does not have every .NET 1.1 API implemented (see the Mono release notes for Mono 1.0) and when executing a binary from Windows that consumes an unimplemented API you might get an obscure message about tokens not being found.

In these cases it is useful to compile your application with Mono's C# compiler just to ensure that you are consuming APIs that are supported.

This is not a perfect solution, as some APIs in Mono throws “Not Implemented Exceptions” in certain cases, so you still should test your application with Mono.

ANP, Second Life amper gebruikt, 2007 [40]

Original article:

RIJSWIJK - Slechts een kleine minderheid van de ruim 300.000 Nederlanders die zich op Second Life hebben geregistreerd, begeeft zich ook daadwerkelijk in de virtuele wereld. Uit cijfers die beheerder Linden Labs op zijn website heeft gepubliceerd, blijkt dat zo'n 17.000 Nederlanders hun figuurtje in Second Life actief gebruiken.

De cijfers bevestigen eerdere indicaties van het bedrijf dat veel mensen een zogenoemde avatar op het internetprogramma aanmaken, maar vervolgens niet meer naar dit alter ego omkijken.

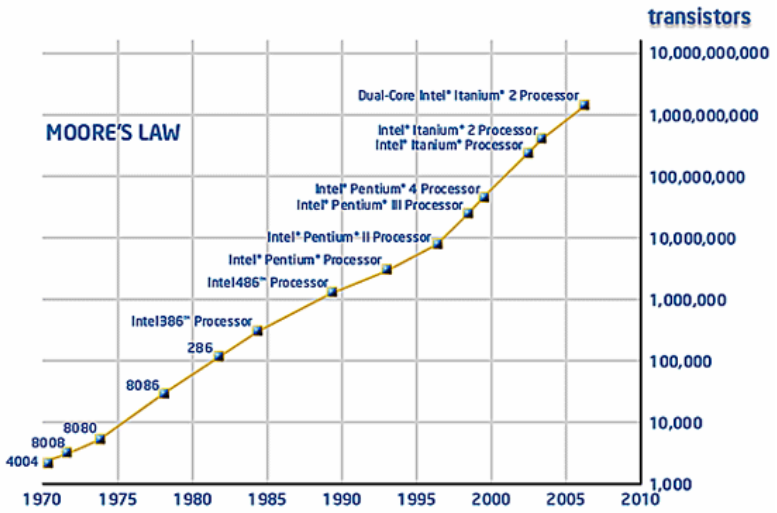
De Nederlandse gebruikers die wel actief zijn, staan wereldwijd op de vijfde plaats als het gaat om de tijd die ze er doorbrengen: gemiddeld spendeerden zij in mei zo'n 52 uur per maand aan het spel. De Amerikanen vormen met 130 duizend gebruikers de grootste groep op Second Life.

Bron: ANP

Excerpt: From numbers, presented on the website of Linden Labs, it can be viewed that from 300.000 Dutch subscribed users, only about 17.000 are really active within second life. Those same numbers state there are only 130.000 active American users in Second Life. It seems many people once create an avatar, but never use it after a first trial. Source: the Dutch Press Agency.

Intel, Moore's Law, 2007 [41]

As stated in the article at http://www.intel.com/technology/mooreslaw/index.htm, Moore's Law is still in effect. It states that the number of transistors on a chip doubles about every two years. According to Intel, this is still true.



APPENDIX B:
Finding a suitable programming environment for the development of a graphical user interface

Introduction

Creating a graphical user interface in Xwindows is something that is very hard without support of a high-level development tool or graphical language. The Xwindows C-libraries are not easy to use and it is very difficult to have a notion of how the look of the system actually will be when programming it.

Most of the first version of the Desktop Tele-classroom Conference system [1] is implemented using C. A small part had been added in the form of a shell-script to make it easy to start and stop the components of the system.

To implement a graphical user interface for the Tele-classroom Conference system it is recommended to use a special graphical development environment. This can consist of a graphical language, a set of tools or code-generators or a mixture of all those.

The criteria to select a development environment are given below:

C-support:

A development environment must support C and so must have at least one of the following properties:

- It consists of some kind of C-super-library with very powerful graphical functions.
- It generates a kind of framework in C-code in which you can put links to your own code.
- The environment / language supports the calling of the necessary C-subsystems in executable forms from within.

B&O Supported:

The Sun/Sparc stations we use for the first prototypes are being maintained by B&O. They should be able to support the necessary programs and libraries for our chosen system/language.

Interactiveness of visual design:

The more interactive a development system is, the more easily and quickly one can design and adapt tot changes and the better standardised a user-interface will be [6].

Level:

The level of a system/language says something about the kind of detail of the interpretation/coding that is automatically performed by the system/language without explicit programming. The higher the level of a system or language is, the less detail the programmer has to worry about and the less room there is for mistakes.

Functionality:

The functionality of a system/language can be seen as the ability to express certain functions in that system/language without looking at the kind of detail with which one has to concern oneself. The more functionality a system/language offers, the more one can actually implement using this system/language without the help of another system/language.

Compatibility:

This is the ability for a system/language to be compiled/executed in different hard/software environments. This should be as high as possible, but at least it should be able to perform on a Sun/Sparc with Sun OS5 (Solaris 2.5)

Maintenance:

The amount of trouble that goes into making small changes and keeping all the components of the whole system (including the c-code) working together whenever one of those small changes occurs.

Efficiency:

This is the resulting speed of the system-code at execution-time. The resulting code must not result in an intolerably low performance because this part of the system has to interact in real time with a human user.

Documentation:

The more and the better documentation is available, the better and faster one can use the full potential of a system/language and the less chance there is for mistakes.

The best way to design a graphical user interface is with aid of another graphical user-interface itself.

The reviewed systems

Devguide

This is a graphical development tool designed by Sun in which you can interactively design the appearance of a user-interface. You first create a main window with some basic widget-containers. After that you can “drag” all kinds of controls into those containers and position them at will. After that, it generates a framework in C to which you can add and/or link your own C code. The two images fig. 1 and 2 give an example of the graphical design interface and a small design using Devguide.

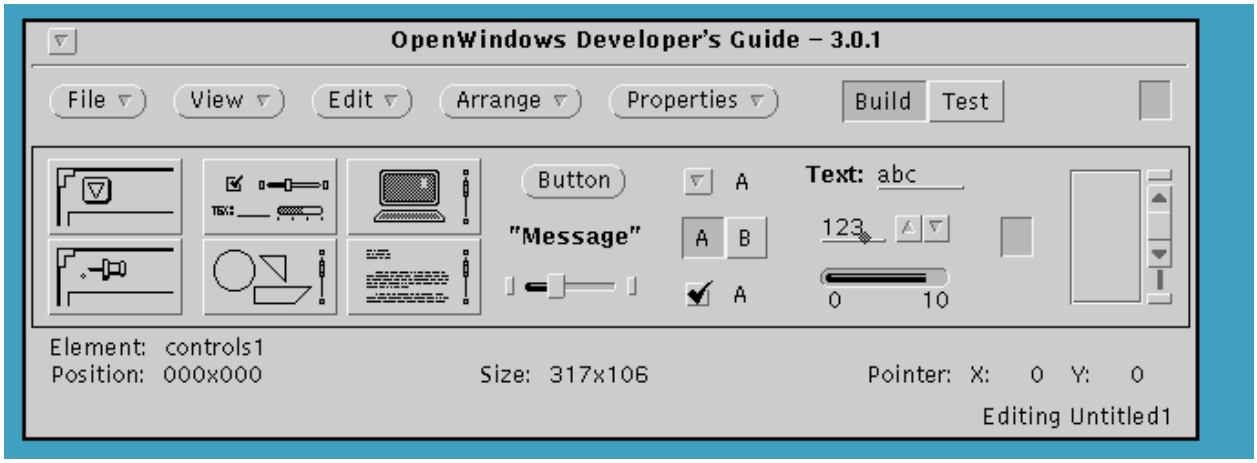


fig. 1 Designer interface.

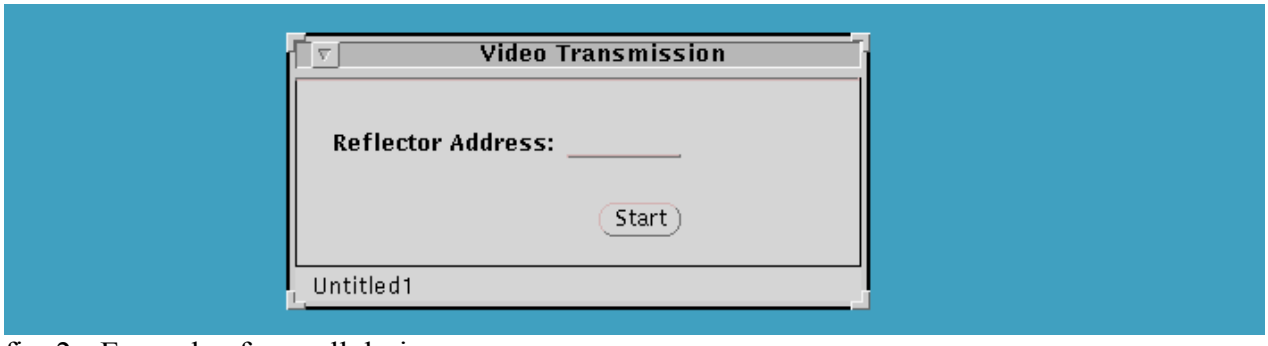


fig. 2 Example of a small design.

Point wise evaluation of Devguide according to the given criteria:

C-support:
It is compatible with C since this tool generates a framework in C-code which has to be completed with user designed functions

B&O supported:
It uses the Xview-libraries. These are rather old and not supported by B&O anymore, so it is risky to use them. These might also not be available anymore for all types of unix-workstations.

Interactiveness of visual design:
It has a true graphical design interface, so you can actually see what your interface is going to look like.

Level:
The level of this development system is moderate. The code for the windows-interfaces is automatically generated. When interactions occur with components of those windows, the accompanying user-designed C-procedures are called and all further actions and graphical relations have to be programmed in C itself.

Functionality:
The functionality of this development system is very high since it is directly in C-code. Everything that is needed can be directly programmed in user-functions which are completely integrated into this system.

Compatibility:
Devguide has only support for the open look version of Xwindows under Solaris. The recompiled C-code might work at other unix-systems provided they do have Xwindows and the Xview-libraries available. There is no support whatsoever for other operating systems.

Maintenance:
Devguide generates the whole graphical system in C-code and keeps the once made user-defined functions. This means that, at the end, you have one C-program that needs to be recompiled and a set of graphical definition-files. Unfortunately this means also that any names given to object can't change without changing the dependent user defined code.

Efficiency:
The code is very speed optimised since it solely consists of compiled C.

Documentation:
A comprehensive manual is present in Sun's Answerbook.

Summary: Devguide has some interesting features but the fact that it uses a number of old library-components which are explicitly not supported by B&O anymore makes it very risky to use. That will probably mean that those libraries will not be available anymore for new types of machines so its compatibility with newer SunOS/Solaris machines is in danger as well. It is also needed to have a good notion of how the automatically generated parts work, to complete the HC-interaction.

Xform 1.0

This tool can generate small forms with widgets. It is called from within shell-scripts or C-programs. The layout of the windows-forms and the actions to be performed on certain events are described in configuration-files. The actions take the form of writing something to a standard output or file or executing a system call.

Point wise evaluation of Xform 1.0 according to the given criteria:

C-support:
: It can be called from within scripts or C programs. Communication about user-interaction can only be done by executing small C programs or by use of a read/write file.

B&O support:
: It uses only standard Xwindows functions, so it should work on all our standard Sun/Sparcs.

Interactiveness of visual design:
: It has no means of graphically designing windows.

Level:
: The level of this development system is high. It is like a kind of simple script language with graphical functions.

Functionality:
: The functionality of this development system is very low, since it only has a very limited set of functions only designed to show a few Xwindows widgets like windows, text-fields and buttons. For instance, there is not something available as a slide bar and there is no means of directly relating widgets within the same window session to quickly show changes for related topics.

Compatibility:
: The system has support for Xwindows under Solaris. It will probably work at other UNIX systems provided they do have Xwindows. There is no support whatsoever for other types of operating systems.

Maintenance:
: The scripts and the c-code are strictly separated.

Efficiency:
: The resulting system is not very fast since it must execute different programs/system calls with each button press, or alternatively, must read/write files all the time.

Documentation:
: None is available except a small readme file.

Summary: Xform 1.0 is very small and relatively easy to use in shell-scripts, but apart from that it has a large number of drawbacks. One of the things it lacks most is more functionality. Even the number of widget-types that are supported is too small for us.

Java

Java is a machine-independent language that is compiled into a sort of intermittent, machine independent code which is run by a "virtual Machine". This is actually a kind of real-time interpreter. It runs on all kinds of UNIX and Microsoft Windows systems. It has very powerful libraries (in the form of classes of objects) that provide good functionality for graphical widgets. It also has libraries that provide functionality for interfacing with network-sockets this according to online documentation presented by Sun Microsystems [10] .

It could be used to implement some, not to processor-demanding, network protocols, but probably not for very demanding operations like MPEG-decoding or real-time audio-processing since, for the moment, it runs only in a "virtual machine" i.e. is interpreted at run-time.

Point wise evaluation of Java according to the given criteria:

C-support:
: Java can directly use C code from dynamically loaded libraries. Unfortunately this feature is still in development and not finalised yet.

B&O support:
: Java is supported by B&O so it should work on all our standard Sun/Sparcs.

Interactiveness of visual design:
: It has no means of graphically designing windows as yet.

Level:
: The level of this development system is high. It has some powerful build-in object classes to deal with graphical user interfaces and networking.

Functionality:
: The functionality of this development system is very high since it has build-in object classes for a variety of functions. It would be possible to implement most of the network protocols in it, apart from the raw audio and video transmission.

Compatibility:
: Java is supported for many versions of UNIX (also for Solaris), Windows95/NT and Macintosh systems. For UNIX there is support for the use of dynamically loaded libraries. For windows95/NT machines support for the use of dynamically loaded libraries is only available if they are programmed with Microsoft Visual C++ version 2.x.

Maintenance:
: All programming structures can be shielded from each other by the use of classes. But it should be noted that Java is still being developed and that there will be some changes, especially when it comes to supporting C. This might cause problems in the future.

Efficiency:
: The resulting system is reasonably fast because it uses an intermitted machine independent code. In the future it should even be possible to compile Java completely into real machine-code.

Documentation:
: Reasonably good documentation is available on the Internet [10] .

Summary: Java has many interesting features, and run in many environments, but at some points the compatibility with other operating systems and languages is not yet optimal and is being subject to future changes. Also there is no current support for graphically designing a user interface.

TCL/TK (7.4 / 4.0 or 7.5 / 4.1) with GuiBuilder 1.0

TCL/TK is a (mostly) machine-independent script-language that is being developed by Sun. It is interpreted at run-time. It runs a number of operating systems. It provides very good graphical and easy functionality.

In combination with GuiBuilder, a graphical design interface for TCL/TK programs that itself is written in TCL/TK, it becomes a very easy to use system for implementing graphical user interfaces. There are also C-libraries available which can perform most of the TCL/TK functions. The two images fig. 3 and 4 give an example of the graphical design interface and a small design using TCL.TK with Guibuilder.

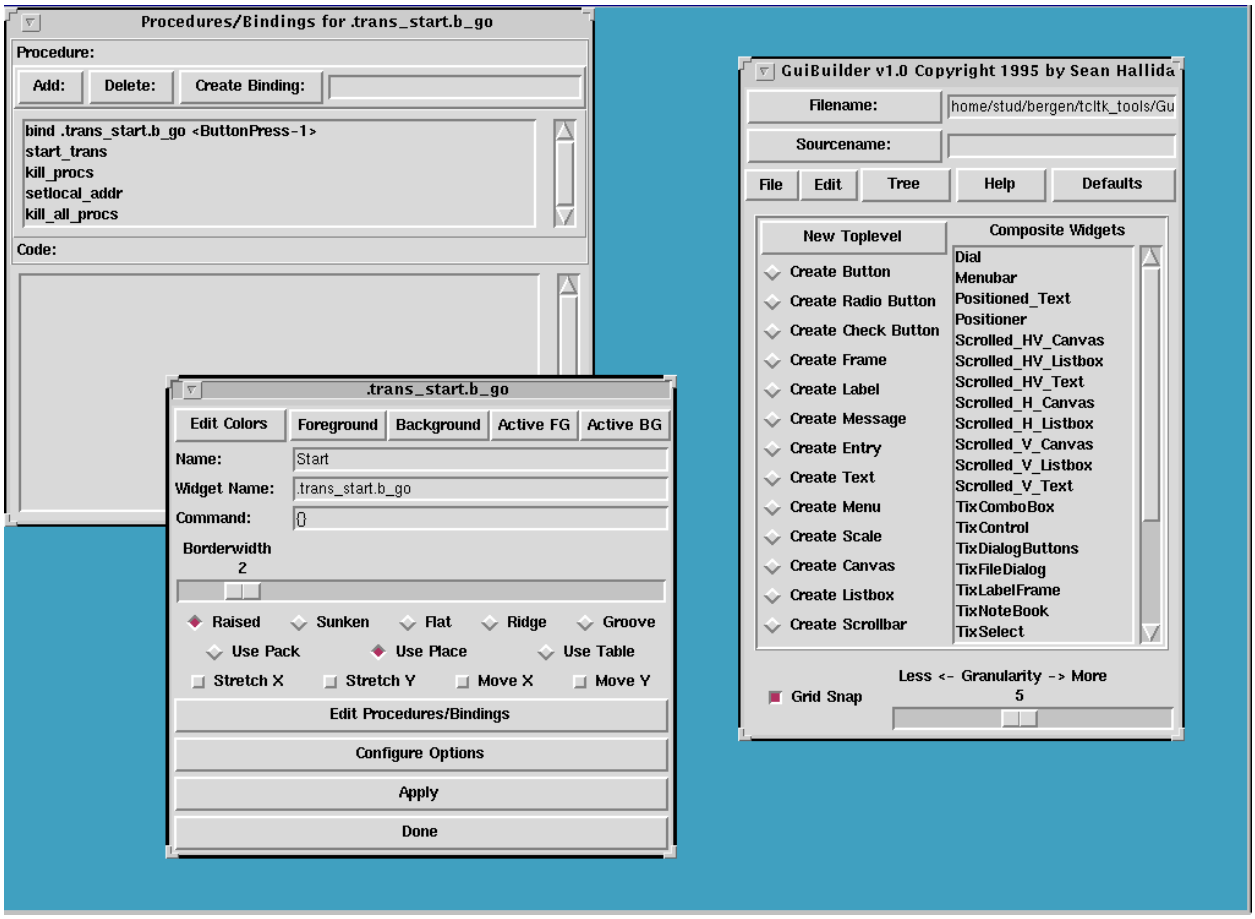


Fig. 3 The designers interface.

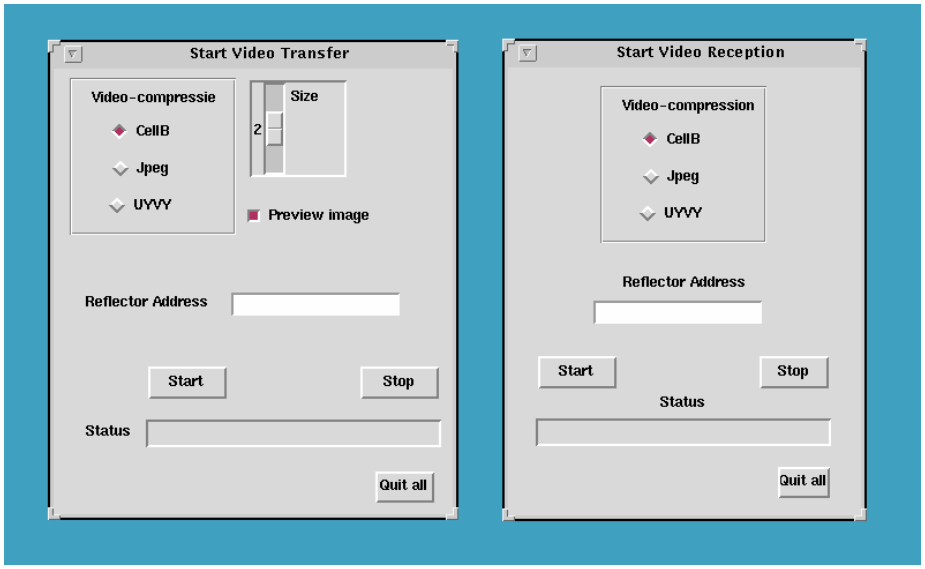


Fig. 4 a sample of a designed user interface with TCL/TK

Point wise evaluation of TCL/TK + Guibuilder according to the given criteria:

C-support:

It can support C in two ways. TCL/TK scripts can be used to execute C-programs and communicate with them using pipes, or the TCL/TK functionality can be used from within C programs in the form of a set of library functions.

B&O support:

TCL/TK 7.4 and 4.0 are supported by B&O so it should work on all our standard Sun/Sparcs.

Interactiveness of visual design:

There is a good interactive design tool available for the script version of TCK/TK 7.4/4.0.

Level:

The level of this development system is high. It is a script language with powerful graphical/interactive features while part of that code can even be automatically generated.

Compatibility:

TCL/TK is supported for UNIX, Windows95/NT and Macintosh systems. In UNIX and windows95/NT, standard C-libraries and include files are available. For Windows95/NT, the TCL/TK libraries themselves are compiled in Borland C++ 4.5, but should work with most C-compilers.

Maintenance:

The TCL/TK-scripts and the c-code are strictly separated and can exchange information only by parameters and pipes. When using the TCL.TK library, the whole system becomes one set of C-source files that have to be recompiled.

Efficiency:

The resulting system is not too fast when being used as a script since it has to be interpreted at runtime. The speed will probably increase when it is used as a C-library.

Documentation:

Good documentation is available on the Internet and in books [11].

Summary: TCL/TK is a very promising language since it combines good functionality with high compatibility. Furthermore, there exists a design tool for the user-interface.

A TOTAL SUMMARY

All properties of all examined systems are now presented in one table and given a value between "---"and "+++"

	C-support	B&O support	Interactive design	Level	Compatibi-lity	Mainte-nance	Efficiency	Documen-tation
Devguide	++	--	++	+	-	+-	++	++
Xform	-	+-	-	+	+-	-	-	-
Java	+	+	-	++	++	+	+	++
TCL./TK	++	+	++	+++	+++	+	+-	+++

This table shows clearly that, for now, Tcl/Tk is the best language to be used to implement a HCI for our Desktop Tele-classroom Conference system.

Furthermore, at Sun Microsystems, the developers of both TCL/TK and JAVA, a document [9] is available that relates TCL/TK to JAVA. In this document, a comparison is being drawn between Tcl/Tk and Visual Basic and between Java and C/C++ in which TCL/TK takes care of the Graphical User Interface and JAVA of the parts that require more efficiency. This document clearly states that people at Sun Microsystems, the developer of both JAVA and TCL/TK prefer TCL/TK as the language to implement graphical user interfaces.

Therefore, the final conclusion is that TCL/TK is the best language at this moment to use for the implementation of the HCI for the Desktop Tele-classroom Conference system.

APPENDIX C : Detailed, point wise lecture scenario and task analysis for the desktop Tele-classroom Conference

Scenario

Teacher T:

Teacher enters electronic classroom.
Teacher identifies himself and the lecture to be given.
Teacher checks presence of students.
Teacher can remove students
Teacher starts lecture
Teacher gives lecture.
Teacher present lecture notes.
Teacher poses questions to one specific or all students.
Teacher answers questions by students.
Teacher interrupts one student
Teacher interrupts all students
Teacher deletes requests for speech posed by students.
Teacher uses local media control to adapt video images and audio.
Teacher ends lecture.
Teacher leaves electronic classroom.

Student S:

Student enters electronic classroom.
Student identifies himself.
Student attends lecture.
Student reads lecture notes.
Student indicates to teacher he has answers or questions
Student poses questions to teacher.
Student answers questions posed by teacher.
Student uses local media control.
Student leaves electronic classroom.

Task analysis for the lecture scenario

The tasks analysis can be seen as a refinement of the scenario.
Given the above lecture scenario, a number of goals and tasks will be identified.
This definition is by no means a procedural one, but gives only an indication to the order and hierarchy of the different tasks that can be performed.
Some tasks are more or less hierarchically organised. This is represented by an indention.

Goals

The goal for the teacher (T) is, to give a lecture
The goal for a student (S) is, to attend a lecture.

Task analysis for teacher (T)

First, the teacher starts the Desktop Tele-classroom Conference application.

The teacher chooses a class.

The teacher identifies himself and the lecture to be given. This consists of two separate inputs:
- An identification (name and password).
- A lecture-name.

The teacher checks presence of students.

The teacher views the class-list.

The teacher compares the class with a subscription-list and note absent students.

The teacher can remove students.

The teacher can chose a student from the class list and can delete student.

The teacher can let all students who are not present in subscription list be automatically removed.

The teacher gives a lecture. This consists of the use of media to actually give the lecture and to directly communicate with students to pose / answer questions. More detailed:

- The teacher uses media to give a lecture. These media can be of the following type:
  - Audio: This can be used to speak or let students listen to a recording.
  - Video: This can be used to show oneself, a video-recording or pictures.
  - Text: This can be used to present lecture-notes and remarks.
  - Whiteboard: A whiteboard can be used to make or show drawings and to present sheets.
- The teacher poses questions to one student
  - The teacher poses the question.
  - The teacher selects a student from class-list and enables speech for that student and wait for an answer.
  - The teacher waits for the student to return speech, or disable speech for that student himself.
- The teacher poses a question in general, to all the students.
  - The teacher waits for a student request to speak.
  - The teacher enables one ore more media for that student.
  - The teacher waits for an answer.
  - The teacher waits for the student to return speech, or disable speech for that student himself.
  - The teacher waits for the rest of the student(s) to remove speech requests for this subject or removes them actively.

The teacher answers questions by students
- The teacher looks at the request-list for question-summaries.
- The teacher selects a student from class-list or request-list and write-enables one or more media for that student.
- The teacher waits for student to finish question.
- The teacher answers the question using one ore more media.
- The teacher waits for the student to return given write rights for the media or takes them back actively.

The teacher interrupts one student with write rights for one or more media. This can be done by selecting student from class-list or request-list and disables write rights for one or all media for that student

The teacher can interrupt all students with write rights to one or more media.
- Disable write rights for one ore more media for all students

The teacher can deny requests for write rights to certain media, posed by students by selecting a student in the class list or the request list and removing the pending request

Local media control
- audio
  - input volume
    - source selection
    - microphone
    - external source (CD, tape, video)
    - mute
  - output volume
    - destination selection
    - speaker
    - headphone
    - mute
- video
  - preview
  - preview size
  - view size
  - transmission size
  - in mute
  - out mute
  - brightness
  - source
    - camera
    - external source (video-recording)

Ends lecture
Leaves electronic classroom

Task analyses for Student (S)

Enters electronic classroom.
Choose classroom (reflector)
- Identifies himself
- Gives name and student-number.

Attends lecture.
- Watch video medium
- Listen to audio medium
- Read text medium
- Look at whiteboard media (not implemented)

- Indicates to teacher he has answers or questions.
- Poses questions to teacher.
  - Speaks in microphone
  - types text
  - sends video
  - uses whiteboard (not implemented)
- Answers questions posed by teacher.
  - Speaks in microphone.
  - Types text
  - uses whiteboard (not implemented)

Local media control.
- audio
  - input volume
    - source selection
    - microphone
    - mute
  - output volume
    - destination selection
    - speaker
    - headphone
  - mute
- video
  - preview
  - preview size
  - view size
  - transmission size
  - in mute
  - out mute
  - brightness
  - source
    - camera
    - external source (video-recording)

Leaves electronic classroom.

APPENDIX D: Scenario description and task analyses for a generic teleconference system

Scenario description

Groupmeeting Controller (GC):

Enters conference and groupmeeting.
Identifies himself and the groupmeeting.
Checks presence of groupmeeting participants and/or floor managers.
Removes groupmeeting participants.
Defines media / floors.
Assigns or changes floor controllers.
Takes part in groupmeeting, using (own) assigned floors
Ends groupmeeting.
Leaves groupmeeting and conference.

Floor Controller (FC):

Enters Floor.
Checks presence of floor participants.
Assigns or changes floor participants.
Grants floor participants access
Denies floor participants access.
Takes part in floor usage which can consist of one medium like audio, video, text and whiteboards.
Grants floor write access
Takes back floor write access
Manages floor.
Local media control.
Ends floor control.
Leaves floor.

Groupmeeting Participant (GP):

Enters conference and groupmeeting.
Identifies himself
Asks groupmeeting controller to make him floor controller for a number of floors
Becomes floor controller for a number of floors
Asks floor controllers to make him floor participant for a number of floors
Becomes floor participant for a number of floors
Takes part in groupmeeting using assigned floors
Leaves groupmeeting and conference.

Floor Participant (FP):

Enters floor.
Looks/listens to media.
Requests media access to the floor.
Returns floor access
Local media control.
Leaves floor.

Task analysis

Given the above teleconference scenario with the 4 known roles, the following tasks can be defined. This definition is by no means a procedural one, but gives only an indication to the order and hierarchy of the different tasks that can be performed.

Goals

groupmeeting controller:	to manage a groupmeeting.
groupmeeting participant:	to attend a conference/groupmeeting
floor controller:	to manage a floor.
floor participant:	to attend a floor.

A) Task analyses for the Groupmeeting controller

1 Enters conference and groupmeeting
- Identifies himself
  - Identification (name/number)
- Chose a groupmeeting
  - set groupmeeting identity

2 Manages groupmeeting
- Checks presence of group participants
- View the subscription-list (not to be implemented yet)
- Compare with subscription-list and note absent participants (not to be implemented yet)
- Removes group participants
  - Chose a participant from the groupmeeting list and delete participant. (not to be implemented yet)
  - Automatically remove all group participants who are not present in subscription-list. (not to be implemented yet)
- Manages floorcontrol
  - Looks at floor control request list for request-summary
  - Selects group participant from request list and give floor control to that participant for one or more floors.
  - Frees a floor of all control.
  - Waits for participant to release floor control.
  - Takes back floor control from that participant for one or more floors.
  - Rejects floor control requests
- defines floors
  - adds floor
    - gives floor identity
    - choose media-type
    - choose controller
  - deletes floor
    - enters identity of floor to be deleted

3 Takes part in groupmeeting
- Requests FP-assignment for one or more floors to the accompanying floor controllers
- Makes himself FC for a number of floors
- Return FP-assignment rights for one ore more floors to floor controller

4 Ends groupmeeting
- Leaves groupmeeting and conference

B) Task analyses for Floor Controller

2 Gets floorcontrol rights

3 Checks presence of floor participants
- View the floor participant list

4 Removes floor participants
Chose a participant from the floor list and delete participant. (not to be implemented yet)

5 Manages floor
- Looks at floor access request list for request-summary
- Rejects floor access requests
- Give floor access to that participant for one or more floors.
- Wait for participant to release floor access.
- Take back floor access from that participant for one or more floors.
- Take back floor access from all participants for all managed floors.

6 Local media control
- audio
  - input volume
  - source selection (only mock-up)
  - microphone
  - external source (CD, tape, video)
  - in mute
  - output volume
  - destination selection
  - speaker
  - headphone
  - out mute
- video
  - preview
  - preview size (not to be implemented yet)
  - brightness (not to be implemented yet)
  - projection size (not to be implemented yet)
  - transmission size
  - in mute
  - out mute
  - source (not to be implemented yet)
  - camera
  - external source (video-recording)

7 Uses floor
- video
  - Watch video
  - Watch preview image
  - look into camera
  - play recording
- audio
  - Listen
  - Talk
  - play recording
- text
  - view text
  - type text
- whiteboard (only mock-up)
  - set size
  - type text
  - draw figures
  - watch contents

8 Releases / loses floorcontrol rights

C) Task analyses for group participant

1 Enters conference and groupmeeting
- Identifies himself
  - Identification (name/number)
- Chose a groupmeeting
  - set groupmeeting identity

2 Takes part in groupmeeting

- Requests FC-assignment (i.e. floor-control rights) to the groupmeeting controller for one or more floors (not implemented)
- Requests FP-assignment for one or more floors to the accompanying floor controllers (not implemented)
- Return FP-assignment rights for one ore more floors to floor controllers
- Return FC-assignment rights for one ore more floors to groupmeeting controller

3 Leaves groupmeeting and conference.

D) Task analyses for floor participant

1 Joins floor

2 Requests access rights for floor

3 get access rights for floor

4 Uses floor (depending on access = read/write rights)
- video
  - Watch video
  - Watch preview image
  - look into camera
  - play recording
- audio
  - Listen
  - Talk
  - play recording
- text
  - view text
  - type text
- whiteboard (only mock-up)
  - set size
  - type text
  - draw figures
  - watch contents

1 Return access rights for floor to floor controller

2 Local media control
- audio
  - input volume
  - source selection (only mock-up)
  - microphone
  - external source (CD, tape, video)
  - in mute
  - output volume
  - destination selection
  - speaker
  - headphone
  - out mute
- video
  - preview
  - preview size (not to be implemented yet)
  - brightness (not to be implemented yet)
  - projection size (not to be implemented yet)
  - transmission size
  - in mute
  - out mute
  - source (not to be implemented yet)
  - camera
  - external source (video-recording)

3 Leaves floor

APPENDIX E: Mapping of task analyses for the generic teleconference scenario to HCI functionality for the DTC system.

A) Task analyses for the Groupmeeting controller	**Functionality for the Groupmeeting controller**
1 Enters conference and groupmeeting	Start-up / login window (HCI-view)
Identifies himself	
Identification (name/number)	Input for name/number
Chose a groupmeeting	
set groupmeeting identity	Input for groupmeeting name/number
2 Manages groupmeeting	Groupmeeting management window
Checks presence of group participants	Present Participants list (show users joining and leaving)
Manages floorcontrol	
Looks at Floorcontrol request-list for request-summary	Present floorcontrol request-list (show new requests)
gives floor control to participant	Command to give floorcontrol to participant.
Frees a floor of all control.	Command to free floor.
Waits for participant to release floor control.	Show floor control release indication.
Takes back floor control	Command to grab floorcontrol from participant
Rejects floor control requests	Command to reject requests for floorcontrol
Defines floors	Floor management window
Adds floor	Command to add floor
Gives floor identity	Input / selection floor-name
Choose media-type	Input/selection for floor type
Choose controller	see 2.5.2 / control defaults to Groupmeeting controller
Deletes floor	Command to delete floor
Enters identity of floor to be deleted	Input /selection floor name

B) Task analyses for group participant	**Functionality for group participant**
1 Enters conference and groupmeeting	Start-up / login window (HCI-view)
Identifies himself	Input for name/number
Identification (name/number)	
Chose a groupmeeting	Input for groupmeeting name/number
set groupmeeting identity	
2 Takes part in groupmeeting	
Requests FC-assignment (i.e. floor-control rights) to the groupmeeting controller for one or more floors (not implemented)	Command to request floor control
Requests FP-assignment for one or more floors to the accompanying floor controllers (not implemented)	Command to request floor participation
Returns FP-assignment rights for one ore more floors to floor controller	Command to return floor participation
Returns FC-assignment rights for one ore more floors to floor controller	Command to return floor control
3 Leaves groupmeeting and conference.	Command to exit groupmeeting

C) Task analyses for Floor Controller	**Functionality for Floor Controller**
2 Enters floor, Gets floorcontrol rights	
3 Checks presence of floor participants	Present Participants list (show users joining and
4 Manages floor	leaving)
Looks for request-summary	Present floor-access request-list (show new requests)
Rejects floor access requests	Offer option to remove/deny requests for floor access
Gives floor access to participant for one or more floors.	Offer option to give floor access to participant.
Waits for participant to release floor access.	Show floor release indication.
Takes back floor access from that participant for one or more floors.	Offer option to grab floor access from participant
Takes back floor access from all participants for all floors.	Offer option to grab floor access from all participants
5 Local media control	
audio	
input volume	Offer input-volume regulator
source selection	Offer source selection
microphone	
external source (CD, tape, video)	
mute	Offer mute option
output volume	Offer output-volume regulator
destination selection	Offer destination selection
speaker	
headphone	
mute	Offer mute option
video	
preview	Offer preview option
brightness	Offer brightness regulator
transmission size	Offer transmission size regulator
in mute	Offer in mute option
out mute	Offer out mute option
source	Offer source select option
camera	Offer camera select option
external source (video-recording)	Offer external source select option
6 Uses floor	
video	
Watch video	Show video window
Watch preview image	Show preview video window
look into camera	Offer camera
play recording	Offer video record play option
audio	
Listen	Offer audio output
Talk	Offer audio input
play recording	Offer audio record play option
text	
view text	Offer text display option
type text	Offer text input option
whiteboard (only mock-up)	
set size	Offer whiteboard sizing option
type text	Offer whiteboard text input option
draw figures	Offer whiteboard figure draw options
watch contents	Offer whiteboard output window
7 Ends floorcontrol	Offer floorcontrol end option
8 Leaves floor	Floor exit function

B) Task analyses for Floor Participant		**Functionality for Floor Participant**
5	Joins floor	
6	Requests access rights for floor	Offer floor-request function
7	get access rights for floor	Show floor access has been acquired
8	Uses floor (depending on access = read/write rights)	
	video	
	Watch video	Show video window
	Watch preview image	Show preview video window
	look into camera	Offer camera
	play recording	Offer video record play option
	audio	
	Listen	Offer audio output
	Talk	Offer audio input
	play recording	Offer audio record play option
	text	
	view text	Offer text display option
	type text	Offer text input option
	whiteboard (only mock-up)	
	set size	Offer whiteboard sizing option
	type text	Offer whiteboard text input option
	draw figures	Offer whiteboard figure draw options
	watch contents	Offer whiteboard output window
4	Return access rights for floor to floor controller	Floor return function
5	Local media control	
	audio	
	input volume	Offer input-volume regulator
	source selection (only mock-up)	Offer source selection
	microphone	
	external source (CD, tape, video)	
	input mute	Offer input mute option
	output volume	Offer output-volume regulator
	destination selection	Offer destination selection
	speaker	
	headphone	
	output mute	Offer output mute option
	video	
	preview	Offer preview option
	transmission size	Offer transmission size regulator
	in mute	Offer in mute option
	out mute	Offer out mute option
	camera	Offer camera select option
	external source (video-recording)	Offer external source select option
6	Leaves floor	Floor exit function

APPENDIX H: The Database for the DTC system

In order to use the DTC-system, there must be a database to determine what conferences there exists, who is who, who may join what conference and has what kind of roles in that conference.

In Appendix B, the database-definition from another application is described [5]. That application might, in the future, work together with the DTC-system. Therefore those elements can act as a guide for the design of the database for the DTC-system.

These elements relations are defined in tuples as follows:

- To determine the correct identity of a user

  Teacher/student ID number + name + password

- To determine the name of a course:

  College ID number + College Name

- To determine what course is being given, by whom, and who may join it:

  College ID number + Teacher ID + List of student ID's

These definitions can be adapted and extended for the DTC system.

The DTC-system needs the mapping from user-id's to user-names so these can stay in a separate database. Furthermore, the same can be said about the mapping from college ID number (i.e. Groupmeeting-ID-number) to college-name (i.e. Groupmeeting-name). That means that colleges and courses can be mapped onto groupmeetings. Password is not used, for the time being, since this is a system, mainly used for testing.

The group meeting (was: course)-definition must be extended to incorporate later versions of the DTC system and not just the situation with one teacher and the rest all students. There can be different groupmeeting types that each can have different kinds of actors.
The choice was made to add the groupmeeting-types for each course and the actor-type for each participant in the course-database and defining those groupmeeting-types and actor-types later in another database.

That last database acts as a "blueprint" for each groupmeeting-type and contains the actual definition of the groupmeeting, determining the number and types of floors, which actors have what roles etc.

The definition for the DTC-system is as follows:

- To determine the correct identity of a user:

  ID number + Name

  This is stored in the file tele_u.ini.

- To determine the name of a groupmeeting:

  College ID number + College Name

  This is stored in the file tele_mn.ini.

- To determine what groupmeeting is being held where, who has what kind of role, and who may join it :

  Group Meeting ID number + GM- type + GM-network location + list of all GM-participants ID's and their Actor-Type

  A default actor name can be used when somebody joins the conference, but has no entry in the groupmeeting database. (E.g. a student that hasn't subscribed for a course but wants to follow it nonetheless) This is defined as the actor-type that belongs to GM-participant number 0, when given.

  This is stored in the file tele_m.ini.

Now, the groupmeeting-types themselves have to be defined, along with their floors and actor-types.

- To determine what kind of groupmeeting type has which floors and which actor-types (groupmeeting and actors definition database):

  Groupmeeting Type
  + list of
  Floor Name + Floor Type
  + List of
  Actor Type + Actor Name
  + list of
  Actor Type + Teleconference role + Floor Types

  These database definitions ensure that the DTC-system will be universally applicable, as far as the database definitions are concerned. However, not all combinations are possible at this moment. This implementation of the DTC-system has many limitations that have their impact on the possible definitions of type, number and use of floors. At the moment, it is only possible to actually define one video, one audio and multiple text floors. Hereby is sink (read) access always enabled so only source (write) access can be defined in this version of the DTC.

  This is stored in the file tele_mt.ini.

APPENDIX I: Changes to the existing implementation of the session layer

In order to use the network layer together with the HCI layer, it was necessary to change the number of colours that was claimed by the video-process. It is now 200, but it might be necessary, in the future, to make it twice as small to gain more colour space for new extensions of the HCI.

It was necessary to implement an extra session-layer message to the HCI to serialize all important video commands to prevent the network layer from crashing:

Video_status: [type] [status]
When a video command is send that will invoke video transmission or reception, the HCI will always wait with the next video command until there is a corresponding video status message. [type] can be "in or "out", [status] can be "initialized", "stopped" or "error". This is to prevent a crash of the underlying video processes when several video commands would be processed simultaneously.

The HCI must now wait for each video-command to complete and issue only a new command after a Video_status primitive has been returned.

Furthermore, a bug has been corrected when a command was given to receive video from a certain address and that address logged of the reflector without actually sending a video image.

Addendum:

The Session-layer software worked more or less stable in a Solaris 5 environment and a working adaptation (at least for video reception and audio transmission) had been made to run in Windows NT 3.51 (See Appendix A). Unfortunately, it does not run anymore in Windows XP or Solaris 8. To adapt this, it should be adapted to the usage of new general libraries (e.g. for TCP/IP communication). Furthermore it should be adapted to use new hardware and new drivers (probably for sound, certainly for video).

Unfortunately it appears that the source code of the old session layer implementation both for Solaris and Windows NT is not available in a complete version anymore. This means a possible adaptation will be difficult and very time consuming since it would require redesigning of many parts of this software.

APPENDIX J: Start-up parameters of the DTC-system

The procedure command-line parameters are parsed when the TCL/TK script is started.

Start-up options are:

-H	: Help message
-L	: no local media
-M nr	: Meeting number
-U n	: User name or number
-Q	: Quick start without login. This requires that a meeting nr and user-nr are given. This will be ignored with the -T option.
-C coding	: Video-coding. Must be CellB, Jpeg, UYVY or Mpeg1. CellB is default.
-N	: No camera available
-S	: Size for video-out. Must be between 0 and 4. Default = 2
-P	: Preview Off.
-F	: Faces disabled (needed because this external module might contain unknown bugs)
-A	: Use ATM
-D	: Debugging-mode on (is default on in test mode)
-T	: Test mode: This means that another start window is used
-E	: Executable for network layer

Addendum:

-X : Emulate the Session Layer. This is done by blocking the start-up of a real session layer. Instead, only a tcl/tk window with buttons that can emulate events coming from the session layer is started. Using this also means there can be no local media.

This was needed because the Session Layer is unusable on current workstations and crashes in both windows XP and Solaris 8.

When the application is started and past the login screen, it causes an extra window to appear where the user can select service primitives to be emulated from the Session Layer as described in [2], Appendix A and in section 4.7. It also enables the user to fill in any necessary parameters for those primitives.

APPENDIX K: Adaptation of Guibuilder V1.0

Adapting versus abandoning

The first prototype of the HCI for the DTC system was implemented in TCL/TK, using a promising, Visual-Basic like design tool, "Guibuilder v1.0". This tool can be used to define and arrange all graphical elements (called "widgets" in Tcl/TK) on the screen, bind procedures to them. Both properties of all widgets and the procedures can be edited later.

The DTC program in TCL/TK grew to demanding and to complex to design it further using Guibuilder V1.0, so the choice was between adapting Guibuilder or stop using it and start using TCL/TK only.

The main advantage of using Guibuilder is that it will be much easier during development now and later in the future to make changes to applications that are designed with its help. Most outward appearance can be designed and changed with much less effort than by using TK's pack-functions, while procedures can be better organised and edited.

Therefore the choice was made to continue using Guibuilder and adapt it to the growing demands.

Adaptation

To make further use of the graphical design tool "Guibuilder" for TCL/TK, it was necessary to get rid of a number of annoying bugs

- Disappearing widgets at random times.
- No standard delete key in editable texts.
- Wrong combining of scrollbars and scrollable widgets.
- Activation of buttons while editing windows, crashing Guibuilder.
- Wrong storage of default values from Guibuilder environment
- It does now correctly work with TCL/TK version 8
- Guibuilder now works also correctly in a MS windows (95/NT) environment.

Furthermore, its functionality had to be extended to incorporate the following functions

- Enhanced list boxes are added that can have individually coloured rows and embedded images.
- Choosing a path for "WISH" in the Guibuilder-defaults instead of using environment settings so its generated programs can easily be started by another user without all environment-settings demanded by Guibuilder.
- Choosing a default path for program-file storage and retrieval is now part of the default-settings.
- Placement of widgets can now be linear connected to the sizing of a window, so they can keep their place relative to the lower-right corner of the enveloping window.
- Sizing of widgets can now be linear connected to the sizing of a container-widget while the upper left corner can be fixed or, as an alternative, the position of a widget can be adapted to a container-resizing without changing its size.
- Creation of windows can now be done dynamically so they can be destroyed as well.
- It is now optional that the windows are all at once created at start-up, so windows can be created and shown later at will.
- There can be multiple, indexed, instances of one designed window. This means that a window-design can be a template for a class of windows instead of only one window
- All Widgets (not only top windows) can be show or hidden at will.
- Colour problems with video windows and different Xwindows displays can be caused by a wrong kind of colour mapping in TCL/TK. This can be corrected within Guibuilder by using an extra default setting for the choice of colour mapping.
- There is a new option in the main window definition to determine an alternate action when a user tries to kill a window.
- The new table geometry-manager from TCL/TK 8 can be used to place widgets.
- There is a new type of main-widget defined that is called a "module". It is meant to be a container for a range of coherent procedures that do not belong to a normal "window".
- The positions for all windows of one application that is designed with help of Guibuilder can be maintained in a separate position-file that can be used to easily store and change default windows-positions and sizes.
- Main windows and modules can be stored and retrieved as "composite widgets" to enable writing modules and widgets separately and combining them later.
- An extra version of the TCL/TK application-code is generated in a SUN environment when Guibuilder saves a file. This version is stripped of all remarks and unnecessary spaces and is therefore slightly faster than the normal version.

Also, a number of enhancements have been made to the editing-environment:

- Multiple procedure edit-windows instead of only one, so comparing and copying becomes easier.
- Fast copy-and-paste option like MS windows has (ctrl+C = copy, ctrl+V = paste) with own "clipboard" instead of mouse drag-and-click selection only. This should also work in all text-widget within programs made using Guibuilder.
- Procedure edit-windows have been enhanced with extra edit functions to automatically use module/widget prefixes, search for strings, replace strings, rename procedures and sort procedure-names.

Guibuilder Procedures / Functions in TCL/TK code

The following functions and procedures of Guibuilder can be useful in TCL/TK code:

wincreate window [nr]	Create an instance of "window" as defined within Guibuilder with number "nr". The name of this window is "window_nr". When "nr" is omitted or 0, the instance is just named "window". When placement-parameters exist for this window, they will be used.
windestroy window	Destroy "window" after storing its placement parameters temporary.
whide widget	Hides a widget after storing its position.
wshow widget	Shows a widget and restores it to its former position.
winicon window	Iconify a window.
set_onkill window command	Executes "command" when "window" is ordered to close.
defaultpos_prepsave window	Save placement-parameters for "window" temporary
defaultpos_prepsave_all	Save placement-parameters for all windows temporary.
defaultpos_restore window fp fs	Restore position of "window". fp and fs are booleans that determine if place and/or size are restored when they were not defined initially in Guibuilder.
defaultpos_restore_all fp fs	Restore all placement-parameters. fp and fs are as above.
defaultpos_save fname	Save placement-parameters for all windows in a file with the name "fname".
defaultpos_load fname	Load placement-parameters for all windows from a file with the name "fname".
fancylistbox name options	Defines an extended list box with the properties of a normal list box, but with the ability to display differently coloured rows and images. It is derived from a normal textbox. If "fb" is such a list box, rows can be changed by using "fb item configure rownumber -option value". For images, the standard textbox-method must be used.

This version of Guibuilder was V1.35.

Addendum: adaptation to Windows XP, Solaris 8 and TK 8.4

Some parameters for widgets were extended in TK 8.4 compared to TK 8.0. A few changes needed to be made to Guibuilder to function properly again. The "–in" parameter is ignored and the index for the "-anchor" parameter was corrected. Also a few references to environment variables were removed, since the user environment both in windows XP and Solaris 8 is another setup compared with earlier environments.

Furthermore, a small tool (ddoc, from M. Schinkmann) that already was present and adapted to Guibuilder in the past to create cross reference indexes in html for all procedures and global variables has been integrated in Guibuilder.

Adaptations have only been done for functions needed in the DTC system. It is not guaranteed that all functions of Guibuilder can be fully used at this time.

The current version of Guibuilder is V1.36.

APPENDIX L: Implementation of the HCI for the DTC system: Description of the main modules and their external procedures used in the HCI of the DTC-system.

Module description: external procedures

Used parameters:

IP-numer:
This is a normal IP-number (format: nr.nr.nr.nr with nr: 0-255).
When a phrase occurs like “at IP-number”, what is actually meant is: “at the computer that has IP-number as Internet-address”.

IP-list
This is a list of IP-numbers.

Floor-identifier:
This is a unique identifier for a floor and consists of a letter determines (mostly) the floor-type and a number, that determines the floor-instance. Types used here are A=audio, V=Video, T=Text.

Media-list:
This is a list of floor-identifiers with an optional direction letter in front each identifier. Direction is expressed as from this application-to-network point of view and can be R=read, W=write, X=both.
E.g. Floors A1 T1, A1 both read and write, T1, only read, would become “XA1 RT1”

Floor_read_db

fr:read_meetingdbase {mynr meetingnr}: boolean
Read conference database-files using personal ID-number “mynr” and meeting ID-number “meetingnr” and defines the groupmeeting accordingly. It stops and returns with 0 if the user/meeting numbers are not found / do not match.
This procedure calls the fh_add_floor procedure for each floor and returns with a 1.
fr:current_mname {} : string
Returns the current meeting name
fr:get_reflector meetingnr {}: number
Return reflector address of meeting with number = meetingnumber.
fr:get_all_meetings_info {}: list
Returns list with alternating numbers and names of groupmeetings in the database.
fr:get_one_meeting_name {meetingnr}: string
Returns the name of the groupmeeting with number = meetingnumber in the database.
fr:set_floor_db {no_mm vid_uncontrol is_teacher}

Define a groupmeeting without reading database-files. “no_mm” is a flag and determines the presence of audio- and video-floors. When this is set, only two text-floors (T1 and T2) are defined. When this is cleared, an audio-floor A1, a video-floor V1 and one text-floor T1, are defined.
“vid_uncontrol” is a flag. When this is set, the video-floor (V1) is not controlled, but free concerning read/write access for each participant.
“is_teacher” is a flag that determines the roles of the current user. When this is set, the user is Floor Controller for all floors that can be controlled. When this is cleared, the user is Floor Participant for each floor.
This procedure calls the fh_add_floor procedure for each floor.

fr:check_for_controller {usernr} : media-list
Returns list of floorid’s for which “usernr” is a Floor Controller (return can be empty).

fr:get_actor_indication {}: string
Return actor indication for the current user in the current meeting.

fr:current_mnr {} :number
Returns current meetingnumber.

Floor_Def

fd:set_fcontroller {media ipnr}: boolean
Register a new floorcontroller for all floors in "media". Set address of controller for “media” to “ipnr”. eturn 1 (true) if this was the first known controller

fd:check_user_is_fc_for_floors {m media} : boolean
Check if current user is Floor Controller for all floors in media-list “media” and returns TRUE if so. M is a flag.
m=0: Give no message.
m=1: Give error message when user is controller for all floors.
m=2: Give error message when user is not controller for all floors.

fd:has_controller {m media} : boolean
Check if all floors in “media” have a Floor Controller present within the groupmeeting and returns TRUE if so. M is a flag.
m=0: Give no message.
m=1: Give error message when a Floor Controller is missing for one of the floors.

fd:get_fcontrollers {media} : IP-list
Returns an IP-list of Floor Controllers for all floors in media-list “media”

fd:own_controlled_floors {media} : media
Returns a media-list in “media” that contains all floor-id’s of floors that are controlled by the current user.

fd:not_own_controlled_floors {media} : media-list
Returns a media-list in “media” of all floor-id’s of floors that are controlled by someone present in the current meeting, but not controlled by the current user.

fd:controlled_by {ipnr media}: media
Returns a media-list of all floors that are in [media] and are controlled by the user at the location given by IP-number “ipnr”.

fd:reset_floors {}
Clears all exisiting floordefinitions.

fd:all_floor_ids {} : media
Returns a list of all defined floor-id’s.

fd:name {floorid} : string
Returns the defined name of “floorid”.

fd:get_floors_controlled_by {c} : media-list
Returns the list of floors, controlled by the controller at IP-number “c”, or the list of floors controlled by type “c”. This type can be empty, U (uncontrolled) or X (controlled by user).
fd:get_floortype {f} : string
Returns the type of the floor with floorid “f”.
fd:id {media} : media-list
Returns a medialist Removes direction indications from “media”.
fd:has_controller {m media}: boolean
Returns true if at least one of the floors in media-list “media” has a controller.
m=0: Give no message.
m=1: Give error message when there is no Floor Controller for all of these floors..
fd:controldir {f} :char
Returns the direction of control for a given floor. Control for a floor can be R (only for read), W (only for write), or X (both directions controlled).
fd:dir {m} :char
Returns the direction of a media-list item. This can be R (read), W (write), X (both) or empty, when there is no direction-letter present.
fd:exists {f} : boolean
Returns TRUE if the floor with floor-id “f” exists.
fd:add_floor {f fl_control_t fl_control_dir_t fl_type_t fl_channel_t fl_name_t fl_w_token_t fl_r_token_t fl_w_mute_t fl_r_mute_t}
Add a floor to the floor-management-layer and give all its parameters an initial value. This proecdure calls the “fu_add_floor” procedure in order to complete the definition of the floor for the floor-use module as well.
fd:channel {f} : integer
Get the defined channel-number for a floor. This channel-number distinguishes data-streams and windows for floors of the same type from each other.
fd:delete_floor {f}
Delete a floor within the floor-management layer.
fd:set_token {dir f value}
Set token for direction “dir” (R or W) of floor “f” to value “value” (0 or 1)
fd:has_token {dir f}: boolean
Returns token status for direction “dir” (R or W) of floor “f” (1 if floor has token).
fd:has_tokens {media}: boolean
Returns 1 if the user has a token for at least one of the floors and directions in “media”.
fd:set_mute {dir f value}
Set mute-status for direction “dir” (R or W) of floor “f” to value “value” (0 or 1)
fd:is_muted {dir f}
Returns mute-status for direction “dir” (R or W) of floor “f” (1 if floor is muted).
d:floors_exists {media-list}: media
Returns all floors in media that actually exist.
fd:all_channels_for_floortype {type} : list
Returns all defined channels for a particular type of floor, given by “type”.
fd:id_from_channeltype {channel-type}: string
Returns a floor-id for a given combination of channel-number and floortype, if one exists. Else, return an empty string.
fd:is_free{f} : Boolean
Returns true if floor “f” is uncontrolled (i.e. is free for read and write access)
fd:user_is_not_controller_for {media}: media
Returns a list of media in “media” for which the user is no controller

fd:set_unusable {dir f}
Set tokens for direction “dir” for floor “f” to 0.

Floor_use

fu:set_mute {dir f}
Set mute-property for a certain direction “dir” and floor “f”. Reception or transmission for floor “f” is henceforward blocked for this floor after calling this procedure.
fu:clear_mute {dir f}
Clear mute-property for a certain direction “dir” and floor “f”. Reception or transmission for floor “f” is henceforward enabled for this floor, when a token is present, after calling this procedure.
fu:clear_token {dir f}
Clear token for a certain direction “dir” and floor “f”. Reception or transmission for floor “f” is henceforward blocked for this floor after calling this procedure.
fu:set_token {dir f}
Set token-property for a certain direction “dir” and floor “f”. Reception or transmission for floor “f” is henceforward enabled for this floor, when a floor is not muted, after calling this procedure.
fu:set_tokens media
Set the tokens for all floors and directions that are given in media-list “media”.
fu:clear_tokens media
Clear all tokens for all floors and directions that are given in media-list “media”.
fu:text_out {channel text}
Send text on text-channel “channel”, when permitted by the write-token and write-mute settings for the accompanying floor. Give an error-message when the write-token is not present.
fu:text_in {channel ipnr text}
Receive and show text from text-channel “channel”, when permitted by the read-token and read-mute setting for the accompanying floor.
fu:enable_video_read {channel ipnr}
Enable reception of video for channel given by “channel” for the transmitter at IP-number “ipnr”. Do this only when permitted by the read-token and read-mute settings for the accompanying floor. This procedure will be called by the vid_in_mgr module.
fu:disable_video_read {channel ipnr}
Disable reception of video for channel given by “channel” for the transmitter at IP-number “ipnr”. This procedure will be called by the vid_in_mgr module.
fu:delete_floor f
This ensures that tokens for floors that will be deleted are first cleared and thus floors are first muted.
fu:redef_floors {m fl_w_token_t fl_r_token_t}
Redefine floors for media “m”. Set tokens to fl_w_token_t en fl_r_token_t when appropriate. This can be used for instance when a controller enters the groupmeeting.
fu:set_audiofloor_volume {channel source_ips value}
Sets the volume for an audio floor given by "channel" to "value" for all sources as given by "source_ips". Or now, since there is only one audio-channel, “channel” is ignored.
fu:add_video_read {channel source_ips}
Enable reception of video from ipnumbers in IP-list “ips”. “channel” is ignored for now.
fu:delete_video_read {channel source_ips}
Disable reception of video from ipnumbers in IP-list “ips”. “channel” is ignored for now.

fu:video_write_unusable {}
This blocks video transmission. This can be called for instance when there is no proper video-hardware present.
fu:video_read_unusable {}
This blocks video reception. This can be called for instance when there is not a fast video card present or when this system is used in a remote session.
fu:video_read_unusable {}
This blocks audio transmission and reception. This can be called for instance when there is no proper audio card present or when this system is used in a remote session.

Floor_control

The following procedures are called from within the part of the HCI that reads the definitions databases or defines a meeting directly and might call corresponding procedures within the network-layer.

fc:floor_free {media}
Defines floors in “media” without control
fc:has_floorcontrol {grasp media}
Set the current user as floorcontroller for “media”, announce this to the rest of the meeting and, if grasp is 1, grasp all floors in “media”.

The following procedures are called from within the user controlled part of the HCI and might call corresponding procedures within the network-layer.

fc:return_floor media
Returns a number of tokens for floors and directions as given in media-list “media”. Reset those tokens within the floor-use module.
fc:grant_floor {media ipnr}
Grants a number of tokens for floors and directions as given in media-list “media” to the Floor-Participant at IP-number “ipnr”.
fc:grasp_floor media
Grasps the tokens for a number of floors and directions as given in media-list “media’ from all floor-participants.
fc:refuse_floor {media ipnr userdata}
Refuse a number of token-requests for floors and directions as given in media-list “media’ to the Floor-Participant at IP-number “ipnr” with a message, given by “userdata”.
fc:grab_floor {media ipnr}
Grabs a number of tokens for floors and directions as given in media-list “media” from the Floor-Participant at IP-number “ipnr”.
fc:withdraw_f_request {media}
Withdraw a request for a number of tokens for floors and directions as given in media-list “media”.
fc:request_floor {media userdata}
Request a number of tokens for floors and directions as given in media-list “media”. “userdate” contains a message to appear together with the request.
fc:floors_not_controlled {media}
Sets floors in “media” without a controller. This can happen when a floorcontroller leaves the groupmeeting.

fc:set_controller {media, ipnr}
: Sets “ipnr” as controller for “media”. This can be called when a grasp or grab floor is received since this can only be sent by a floorcontroller.

The following procedures are called from within the network-layer and call corresponding procedures within the HCI.

fc:add_floor_user {ipnr uid}
: Shows the adding of a groupmeeting- and floor-participant (or controller) at IP-number “ipnr” with name or user-number “uid” to the groupmeeting.

fc:delete_floor_user ipnr
: Shows the deletion of a groupmeeting- and floor-participant (or controller) at IP-number “ipnr” from the groupmeeting.

fc:floor_cancelled {media ipnr}
: Shows the cancellation of a token-request for floors and directions as given by media-list “media” from the floor-participant at IP-number “ipnr”.

fc:floor_returned {media ipnr}
: Shows the return of the tokens for floors and directions as given by media-list “media” from the floor-participant at IP-number “ipnr”.

fc:floor_grabbed media
: Show a token-grasp for floors and directions as given by media-list “media” from the floor-controller at IP-number “ipnr”. Reset those tokens within the floor-use module.

fc:floor_requested {media ipnr userdata}
: Shows the token-request for floors and directions as given by media-list “media” from the floor-participant at IP-number “ipnr”.

fc:floor_granted {media ipnr}
: Shows the token-grant for floors and directions as given by media-list “media” from the floor-controller at IP-number “ipnr”. Set those tokens within the floor-use module.

fc:floor_grasped media
: Shows the token-grasp for floors and directions as given by media-list “media” from the floor-controller at IP-number “ipnr”. Reset those tokens within the floor-use module.

fc:floor_controlled {media ipnr}
: Shows that the Groupmeeting Participant at IP-number “ipnr” now is controller for all floors and directions as given in media-list “media” and change the controller-properties for those floors in the floor_def module..

fc:floor_refused {media ipnr userdata}
: Shows the refusal of a token-request for floors and directions as given by media-list “media” from the floor-controller at IP-number “ipnr”. Show the accompanying message in “userdata”.

fc:floor_was_grabbed {media ipnr}
: Shows the token-grab for floors and directions as given by media-list “media” from the floor-controller at IP-number “ipnr”. Reset those tokens within the floor-use module.

fc:floor_given {media ipnr}
: Shows the token-grab for floors and directions as given by media-list “media” from the floor-controller at IP-number “ipnr”. Reset those tokens within the floor-use module.

Floor_HCI

fh:create_floor_win {type channel name}

Create a new window, if needed, for a specific channel and floor-type (“channel” and “type”) with “name” as window-name..

fh:destroy_floor_win f

Destroy a window for a specific floor (properties of the floor define channel and type).

fh:set_message {dir f msg}

Set message “msg” in the floor-status bar for floor “f” and direction “dir”.

fh:set_init_message {f c dir}

Set the initial message for floor “f”. This must be called, whenever the control-properties of a floor change.

fh:add_floor {f fl_control_t fl_control_dir_t fl_type_t fl_channel_t fl_name_t fl_w_token_t fl_r_token_t fl_w_mute_t fl_r_mute_t}

Add a floor with a number of default parameters. This procedure adds the floor for both the HCI and the floor-control layer.

fh:delete_floor f

Deletes the floor with identifier “f” both within the HCI and the floor-control layer

fh:delete_all_floors {}

Deletes all floors both within the HCI and the floor-control layer. This can be used both during startup and shutdown of the groupmeeting.

fh:set_messages {media msg}

Set a message for all floors and directions as given by media-list “media”.

fh:set_unusable {dir f}

Local media HCI-elements for direction “dir” of floor “f” must be hidden.

fh:get_full_floormanes {media}

Get full floornames for all floors in “media”

users_db

ud:add_to_meetinglist {ipnr name nr}
ud:clear_meetinglist {}
ud:delete_from_meetinglist ipnr
ud:get_current_username remark
ud:get_username {n remark}
ud:get_usernumber inp
ud:is_in_meetinglist ipnr
ud:name_in_meetinglist {ipnr givenr}
ud:read_user_database {}
ud:set_current_username {name nr}
ud:use_no_userdatabase {}

floor_req_grant_db

rg:add_granted_floors f
rg:add_requested_floors {}
rg:add_to_grantlist {media ipnr}
rg:add_to_reqlist {media ipnr userdata}
rg:clear_granted_floors {}
rg:clear_grantlist {}
rg:clear_reqlist {}
rg:clear_requested_floors {}
rg:delete_from_grantlist {media ipnr}
rg:delete_from_reqlist {media ipnr}

rg:get_granted_floors {}
rg:get_requested_floors {}
rg:get_ud_request ipnr
rg:granted_media ipnr
rg:is_in_grantlist ipnr
rg:req_or_all_media ipnr
rg:requested_media ipnr
rg:set_granted_floors f
rg:set_requested_floors f

Network_layer

nl:add_video_source ipnr
nl:broadcast_text {ch text}
nl:broadcast_textcontrol c_text
nl:clear_video_waits {}
nl:delete_video_source {ipnr real}
nl:disable_all_video_read {}
nl:disable_au_read {}
nl:disable_au_write {}
nl:disable_vid_write {}
nl:disable_video_read ipnr
nl:enable_all_video_read {}
nl:enable_au_read {}
nl:enable_au_write {}
nl:enable_reception {}
nl:enable_vid_write {}
nl:enable_video_read ipnr
nl:eval_incoming inp
nl:grab_floor {media ipnr}
nl:grant_floor {media ipnr}
nl:grasp_floor media
nl:init_vid_coding vcod
nl:m_compress media
nl:m_expand media
nl:prepare_vid_out_props {v p}
nl:refuse_floor {media ipnr userdata}
nl:request_floor {media ipnr userdata}
nl:return_floor {media ipnr}
nl:set_mixing_volume {ipnr value}
nl:set_vidr_auto vi_auto
nl:signal_video_ready {direction operation}
nl:start_network {n refl a_atm}
nl:stop_netlayer {}
nl:video_wait_write command
nl:withdraw_request {media ipnr}

system

sy:audiofailure {}
sy:check_address {refl_adr atm_adr}

sy:close_db_file f_id
sy:def_images {}
sy:define ex
sy:emergency_stop {m recover message}
sy:get_atm_adr e_adr
sy:get_audioprops {}
sy:get_cmdln_params {}
sy:get_eth_adr adr
sy:get_local_adr {}
sy:init_atm_conversion {}
sy:init_audio {}
sy:is_windows {}
sy:kill_afters {}
sy:kill_procs {}
sy:open_db_file name
sy:pipe_read single
sy:pipe_write s
sy:quit_all {}
sy:read_db_line f_id
sy:reset_blockingcount {}
sy:set_audio_in_source in_source
sy:set_audio_out_source {speaker head aux}
sy:set_volume_in vol
sy:set_volume_mon vol
sy:set_volume_out vol
sy:start_netlayer {user_id refl_adr atm_adr}
sy:stop_netlayer {}
sy:stophci {}
sy:write_wait {}

Main

main:give_usage {}
main:save_winpos {}
main:status_message s

tele-classroom conference window

Access
w_fc:access:change_trlight {}
w_fc:access:controller_known media
w_fc:access:controller_unknown {}
w_fc:access:grabbed {real media text}
w_fc:access:granted media
w_fc:access:init {}
w_fc:access:rejected {media ipnr userdata}
w_fc:access:requested media
w_fc:access:return {}
w_fc:access:returned media
w_fc:access:set_image img
w_fc:access:set_tlight color

w_fc:access:withdraw {}
w_fc:access:withdrawn media

Menu
w_fc:menu:floor_controller_known {}
w_fc:menu:floor_controller_unknown {}
w_fc:menu:floor_granted {}
w_fc:menu:floor_requested {}
w_fc:menu:floor_return {}
w_fc:menu:floor_withdrawn {}
w_fc:menu:is_controller c
w_fc:menu:tfloor_added {channel name}
w_fc:menu:tfloor_deleted channel
w_fc:menu:user_added {name ipnr}
w_fc:menu:user_deleted ipnr

Participants list
w_fc:plist:add ipnr
w_fc:plist:clear {}
w_fc:plist:define_menu control
w_fc:plist:delete ipnr
w_fc:plist:get_ipnr_selection {}
w_fc:plist:selection_grab {}
w_fc:plist:selection_grant {}
w_fc:plist:selection_refuse {}
w_fc:plist:selection_refuse_arg {}
w_fc:plist:selection_vid_start {}
w_fc:plist:selection_vid_stop {}
w_fc:plist:selection_volume vol
w_fc:plist:set_props {ipnr marksign color}

Request list
w_fc:rlist:add ipnr
w_fc:rlist:clear {}
w_fc:rlist:define_menu control
w_fc:rlist:delete {media ipnr}
w_fc:rlist:get_ipnr_selection {}
w_fc:rlist:selection_grab {}
w_fc:rlist:selection_grant {}
w_fc:rlist:selection_refuse {}
w_fc:rlist:selection_refuse_arg {}
w_fc:rlist:selection_volume vol
w_fc:rlist:set_props {ipnr marksign color}

w_fc:add_participant {ipnr uid}
w_fc:add_request {media ipnr userdata}
w_fc:color_darken {color percent}
w_fc:controller_status_change media
w_fc:conv_singel_col {col factor}
w_fc:delete_participant ipnr
w_fc:first_initialisation {scommand do_debug}

w_fc:floors_canceled {media ipnr}
w_fc:floors_grabbed {real media text}
w_fc:floors_granted media
w_fc:floors_refused {media ipnr userdata}
w_fc:floors_returned {media ipnr}
w_fc:grab_floors {media ipnr}
w_fc:grab_floors_done {media ipnr}
w_fc:grant_floors {media ipnr}
w_fc:grasp_floors {}
w_fc:kill_windows {}
w_fc:mark_granted {media ipnr}
w_fc:refuse_floor {media ipnr userdata}
w_fc:refuse_popup {media ipnr}
w_fc:request_deleted {m_left ipnr}
w_fc:request_floor {media userdata}
w_fc:return_floor media
w_fc:set_listmenus control
w_fc:show_faces {}
w_fc:show_listmenu {lbox x y}
w_fc:snew_floorcontroller {first media c_ipnr}
w_fc:start pict
w_fc:stop_conference {}
w_fc:stop_hci {}
w_fc:vidw_in_hide {ipnr media}
w_fc:withdraw_floor media

text-board window

w_txt:clear_entry channel
w_txt:clear_textdisplay nr
w_txt:create {channel name}
w_txt:destroy channel
w_txt:display_incoming {channel ipnr text}
w_txt:displaytext {channel from text color}
w_txt:hide channel
w_txt:in_message {channel text}
w_txt:out_message {channel text}
w_txt:send_entry channel
w_txt:sendtext {channel text}
w_txt:show channel
w_txt:show_all {}
w_txt:winname channel

local media control window

lm:ai:enable {}
lm:ai:set_message text
lm:ai:set_source {}
lm:ai:show_source {speaker phone aux}
lm:ai:show_volume vol
lm:ai:toggle_frame {}

lm:ao:enable {}
lm:ao:set_message text
lm:ao:set_source {}
lm:ao:show_source source
lm:ao:show_volumes {vol_s vol_m}
lm:ao:toggle_frame {}
lm:ao:use_mute {}
lm:ao:use_speak {}
lm:vi:enable {}
lm:vi:set_message text
lm:vi:toggle_frame {}
lm:vo:change {}
lm:vo:disable {}
lm:vo:enable {}
lm:vo:set_message text
lm:vo:set_vsize dummy
lm:vo:set_vsizebar s
lm:vo:toggle_frame {}
lm:exit_win {}
lm:frame_expand frame
lm:frame_hide frame
lm:frame_shrink frame
lm:hide_floors {dir m}
lm:hide_win {}
lm:init_win {half_dupl s p}
lm:recover_all {}
lm:show_floors {dir m}
lm:show_muting {}
lm:show_win {}

Floor information window

w_fi:destroy {}
w_fi:hide {}
w_fi:init {}
w_fi:list:add f
w_fi:list:clear {}
w_fi:list:del f
w_fi:list:set_props {f name ctrl rd wr ctrl_r rd_r wr_r}
w_fi:show {}
w_fi:show_floor_info media

Session layer emulation window

w_sle:FC_Reject_ind
w_sle:FC_Tokengrab_conf
w_sle:FC_Tokengrab_ind
w_sle:FP_Tokenplease_ind
w_sle:FP_Withdraw_ind
w_sle:Tokengive_ind
w_sle:adding

w_sle:deleting
w_sle:text_stream_ind
w_st1:atm_set
w_st1:atm_unset

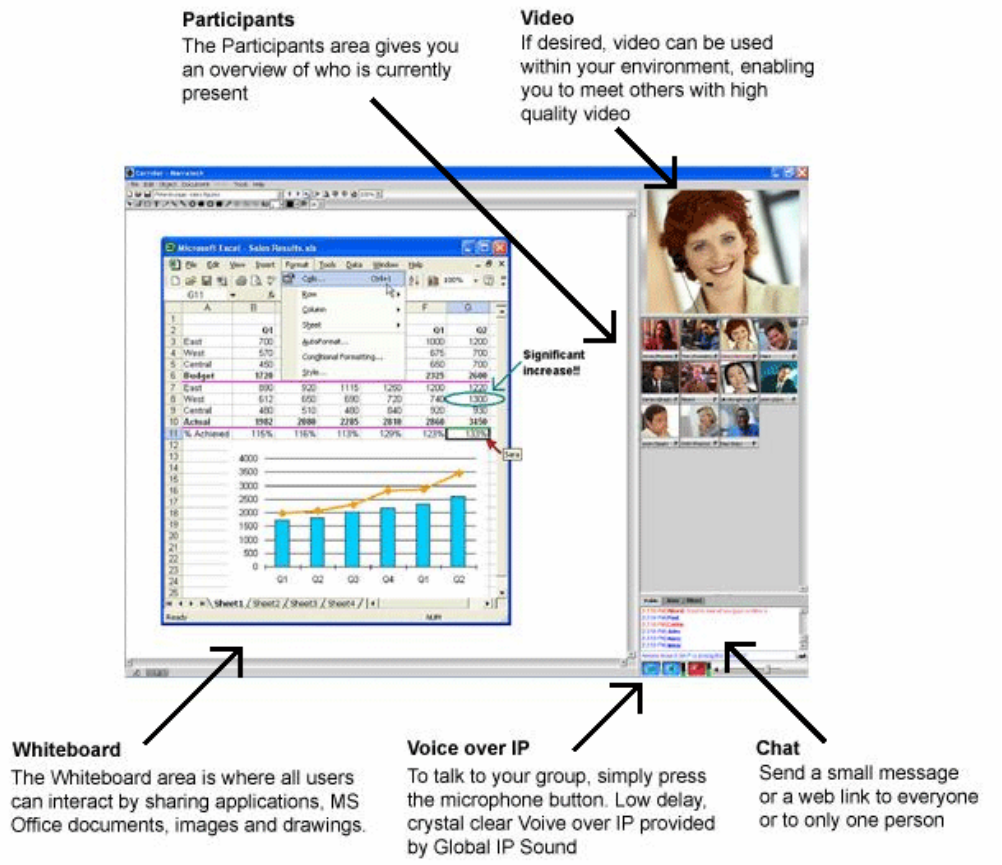
APPENDIX M: Examination of some existing teleconference applications

Marratech

The Marratech 6.0 and 6.1 teleconference system exists of a free downloadable client and a commercially (although there is a free very restricted server version) exploited server, the Marratech manager. Its client and server software is made available for a number of platforms (e.g. Windows, Linux, and Mac OS X) mostly due to the fact that much of the software is written in Java.

On the server side, the Marratech software is compatible with e.g. SIP and H323, making it possible to invite user of other (hardware based) teleconference systems to a teleconference.

The Marratech client enables users to send and receive video, audio, and text. Also it has a one shared whiteboard available. Furthermore, it enables application sharing by using VNC, making it possible for users to view a live application on one system and even handing control of that application over to another user. The standard user interface consists of one window, which is divided in different areas



The Marratech client interface

Teleconferences in the Marratech software are organised in virtual meeting rooms. In each meeting room, users can have one of 4 different roles:

- Moderator: has unrestricted media-access, the ability to kick users and the ability to prevent new users from entering a meeting room. He / she moderates the roles of the rest of the users in a meeting room.
- Presenter: Has unrestricted media access, but no moderation options)
- Attendee: Can only send Public audio, video and text and can use the tele-pointer to point on the shared whiteboard.
- Listener: Has no possibility to send information but can only view and listen to the meeting.

Only the moderator can see the role of other users and there is no mechanism other than using the teleconference media, if possible, to request a role-upgrade (or downgrade). There is a possibly for private text and audio from one user to another user, but there is no option to define more text, audio, video or whiteboard floors for discussions in separate groups or to separate some subjects from one other within the same meeting. If needed, these should be held in different meeting rooms. There is no separate floor request mechanism present.

Adobe Acrobat Connect

This commercially available teleconference system was formerly known as Adobe breeze (or before that as Macromedia Breeze). Adobe Connect uses a current web browser in combination with the Adobe Flash player as its software platform, making it capable (especially for the client) on many systems using audio, video and text communication, although windows and Apple OS X systems have some extra possibilities like file uploading and screen sharing.

Acrobat Connect organises meetings into virtual meeting rooms. Each user in such a room can have one of tree roles:
- Host: Can set up a meeting and invite presenters and participants, add content to the shared library, can broadcast audio, video, can use text chat and can share content. He/she can promote participants to presenter or can give enhance permissions to participants. Also a host can organize participant-polls.
- Presenter: Can broadcast audio, video, can use text chat and can share content.
- Participant: Can use text chat and can view and listen to broadcasted audio, video and content.

The role of each user is visible in the user-list and users do have a status-icon they can set indicating for instance they have a question. The host can rest that status at will.

Acrobat Connect has some means for floor control: As indicated above, the host can promote participants to presenter and vice-versa. Also the host can give participants enhanced permissions for certain media or take these away. This can e.g. be done in response to a question-status set by a user, although status showing and resetting is done independent of assigning roles or permissions.

Tandberg hardware-based teleconference system (used e.g. by Expert at a distance)

The Tandberg (770 / 880/ 990 family) system was demonstrated during a meeting from Glaslokaal, to introduce the ‘expert at a distance’ concept to a number of school representatives in The Hague. Since the organisation I work for (KIVI NIRIA) is also involved with Glaslokaal, I was invited to attend this meeting.

This is an independent hardware based system providing users with a remote controllable pan-tilt-zoom camera and a build-in room microphone. Up to 4 systems can be connected using the H323 protocol without the use of a server to provide a teleconference system. It provides users with a one large view of a video transmission, accompanied with the audio combined with this transmission. A small view of each of the video streams of the participants, including their own transmitted image is also available. By choice, audio and video can be transmitted from an auxiliary input (e.g. a laptop or a DVD-player) instead of the build-in camera.

During a teleconference, no user has any explicit assigned roles. Locally, a system can be controlled by an infrared remote. This makes it possible to select the main image shown on the local screen and to select the source of the image signal that is transmitted to others. It also enables control of the pan/zoom settings of the local camera. For control from another location, it is possible to log in remotely and use a web-interface to control camera settings.

This system is more aimed at lecturer-to-group or group-to-group situations. For everything but direct audio and video, a PC is still needed and pan-tilt options are mostly not a requirement for single desktop users.

Microsoft live meeting

This is a hosted application. It is more an extended collaboration tool, making it possible to show presentations, to use text chat and to organize a meeting. Voice connections are done using external VOIP connections from e.g. BT but can be controlled from within live meeting. It provides 3 roles for users: Presenter, active presenter and attendee. It is possible to promote attendees to presenter, but then can't be demoted in the same session. Apart from giving an attendee a presenter-role, there is separate limited floor control possible for text chatting.

Dimdim Webmeeting

This is an open source webmeeting teleconference application that is still in an alpha phase. It is using the Flash plugin from Adobe on the client side and it uses Java on the server side.

It has options for communicating using audio, video and text chatting. It has no shared whiteboard (yet), but can show documents like PowerPoint or word that have been uploaded (in advance or during the meeting). Furthermore, it has the possibility to share the desktop.

Dimdim has for now only two roles for users: Presenter and attendee. A presenter can communicate with other users using audio, video and text chat and can also upload documents to show or share his/her desktop. An attendee can communicate with the others using audio, video and text chat, but can't share documents or his/her desktop. It lacks many features for dynamic floor control for all media and has no floor request mechanism.

A very interesting feature is that Dimdim provides an integration-pack (still very experimental) for Moodle (an open source web-based e-learning platform).

Vmukti

This is an open source webmeeting teleconference application that was formerly known as 1Videoconference. It is build using .NET 3.0 and is Windows 2000/XP only. For VOIP/video connection it relies on a standard SIP service like Asterisk that needs to be installed separately.

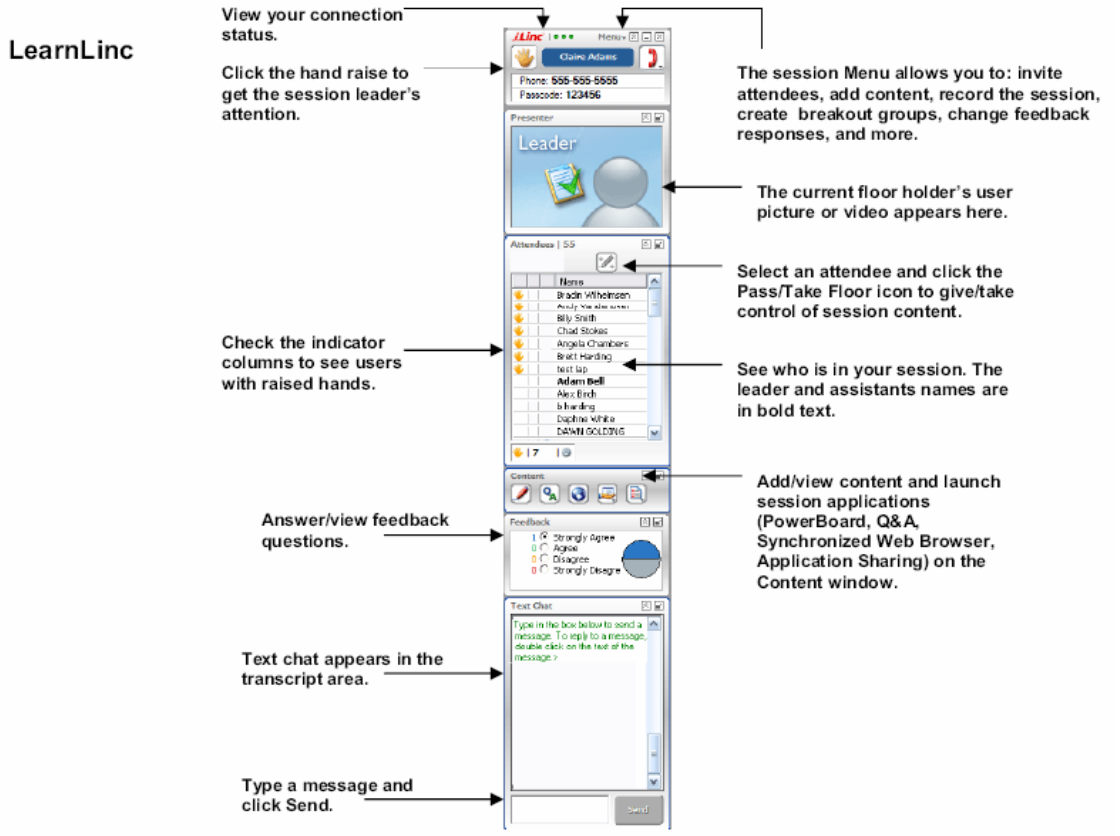
Vmukti has options for communicating using audio, video and text chatting, shared whiteboard, shared applications, and shared desktop. It lacks many features for dynamic floor control and has no floor request mechanism.

A very interesting feature, like Dimdim, is that also Vmukti provides an integration option for Moodle (an open source web-based e-learning platform).

LearnLinc

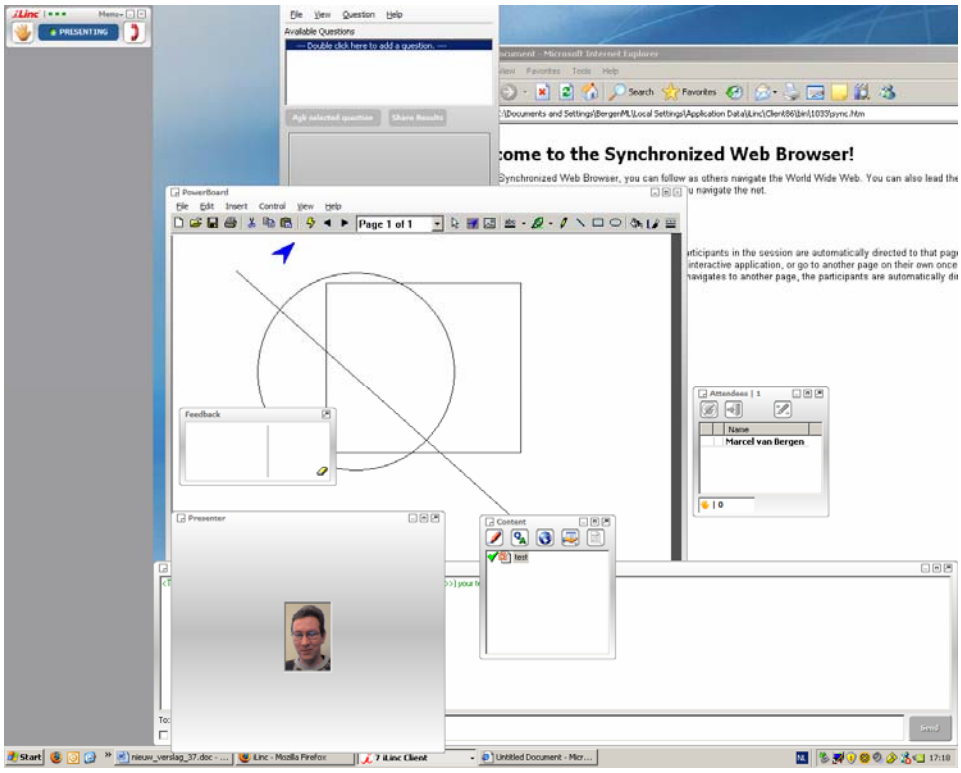
LearnLinc is part of a suite of teleconference applications from iLinc. Other parts of this suite consist of MeetingLinc, ConferenceLinc, and SupportLinc. A previous version of LearnLinc was available in Dutch as Dijdidact and was used at the Open University in Heerlen. All these applications look and work much the same, but in this report, only LearnLinc will be reviewed. They differ only in some small aspects. The iLinc applications run only on PCs running Windows 2000 or XP. ILinc applications use a browser (IE or Mozilla) to login to a teleconference session using a login page, the communications centre. This can contain several meetings and schedules for meetings.

Standard options include document sharing (e.g. PowerPoint), desktop sharing, browser sharing, a shared whiteboard, feedback questions, multiple choice questions lists and surveys.



LearnLinc is organized as a toolbar on the left of the screen with regions (see picture above). It is always visible. The width and length of the toolbar can't be modified, but it can be collapsed to a small mini-bar showing only the connection status, the menu and the hand-raise button.

The separate areas have a fixed order and a fixed size on the toolbar. They can be collapsed into a small indicator-bar on the toolbar or can be undocked and expanded into separate windows that are placed on screen. They can be freely placed and sized (see picture below).



Some integrated parts, the Powerboard (a shared whiteboard) and the Q&A window, have no toolbar-docking mode at all. They can only be used undocked as separate windows.

Standard there is only an option for audio using telephone conferencing available. Audio- and videoconferencing using the internet is optional and requires additional modules.

In LearnLinc, there are three roles for users: leader, assistant and participant.
The leader controls the whole session. An assistant can do everything the leader can do except the following: Grade participants, create breakout groups and speak when he or she does not has the floor. A participant participates in the session and can ask floor control to the leader.

There are means for floor control present in LearnLinc. The leader takes the floor automatically when he or she joins the session. The leader can pass the floor to any participant and take the floor back at any time. While the participant has the floor, he or she can talk and launch content and applications. The leader's microphone is always on, and he or she can talk and launch content even when someone else has the floor.

Participants can request the floor by "raising their hand" (pushing the "raise hand" button by left-clicking). The leader sees a raised hand icon next to the name of the participant in the participants-list and sees raised-hand counter increased by one. He or she can acknowledge that

by giving the floor to that participant or acknowledge this verbally. Giving the floor is done by clicking with the right mouse button on the participants name in the participants list. There is an option to force the lowering of all raised hands. A user can lower his hand by clicking the raise hand button again. Giving the floor is done by double clicking on the specified user name in the attendee-list.

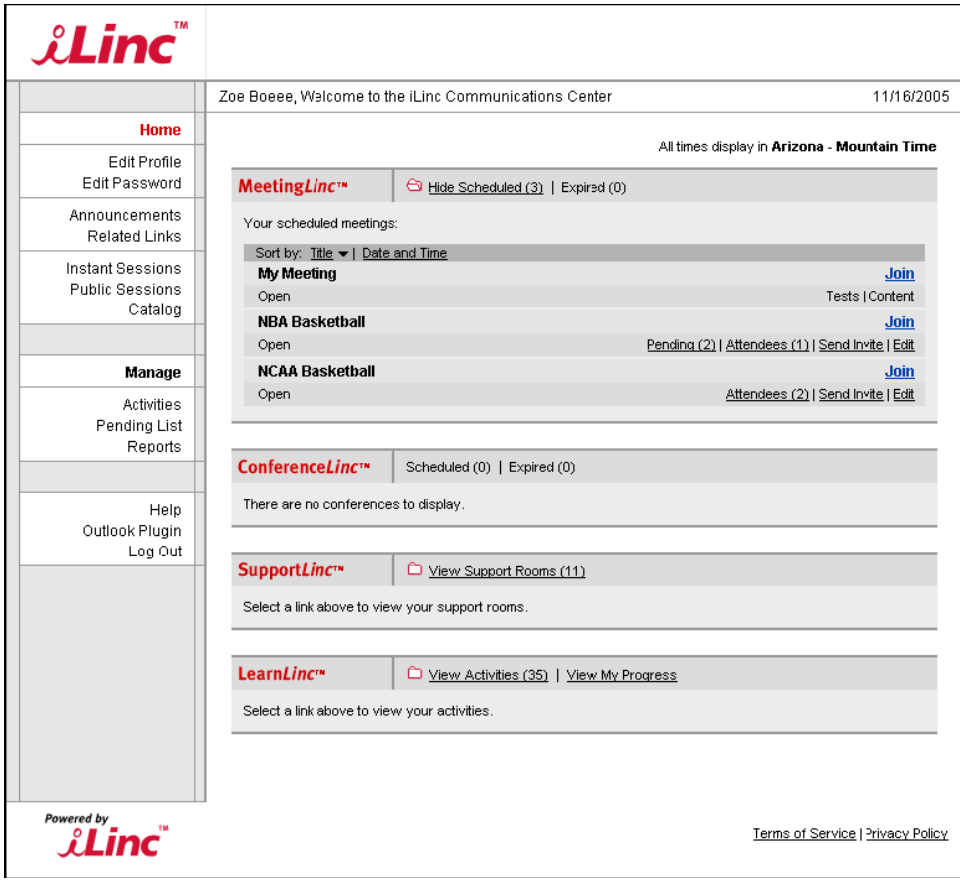
Text chat is not part of that floorcontrol mechanism. A participant in a LearnLinc class can always send messages to all participants at once or to one participant in particular. There is however a separate option for the leader to limit text chat to the ability to send messages to the leader, assistants, and the current floor holder.

In addition to the central meeting (class in LearnLinc) there is an option to organize breakout groups. This is a meeting room to which the session leader sends a group of participants to work collaboratively until they decide to return or for a set period of time. In the breakout group, anything is possible that can be done in the main session, including sharing content, applications etc. There is no floorcontrol during a breakout session. When the breakout group time limit elapses, all participants in the group automatically return to the main session.

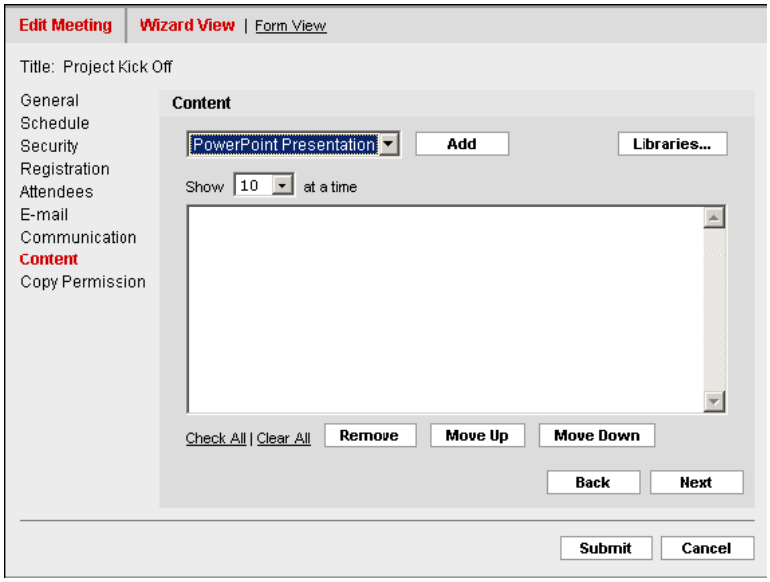
In LearnLinc there are options for the leader and assistants to track the participants engagement (a kind of activity-meter that indicates a participant is active during the last minutes on his / her system or not). He or she is also able to look at the desktop of a user without permission (a function called glimpse).

A leader can assign, edit and view grades for all participants during or after a class session. Participants can view those grades from within the communications centre.

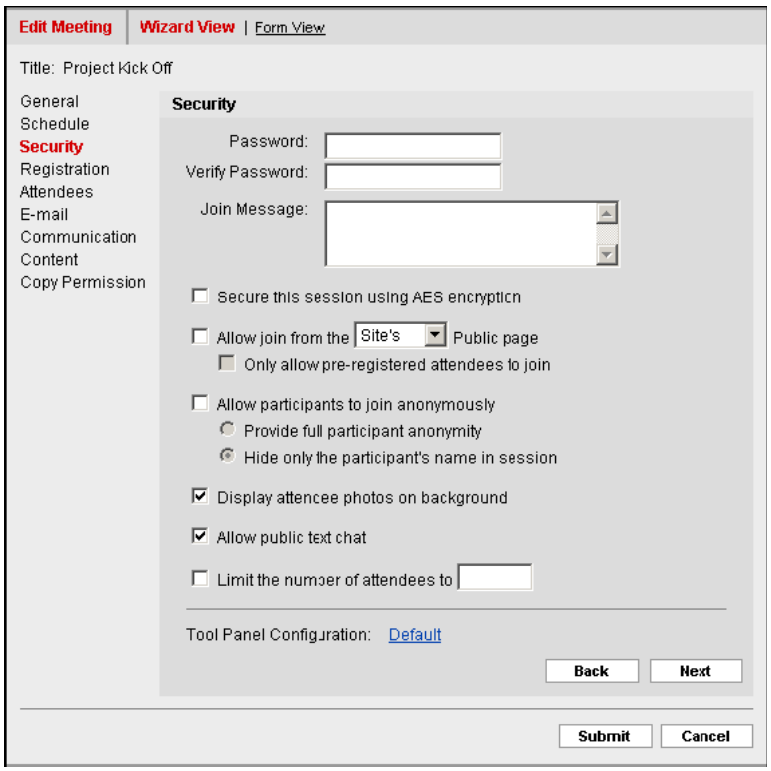
The communications centre is also where the meetings are defined (see below). This means meeting schedules, predefined content, participants, communications settings, security settings and more can be defined from here.



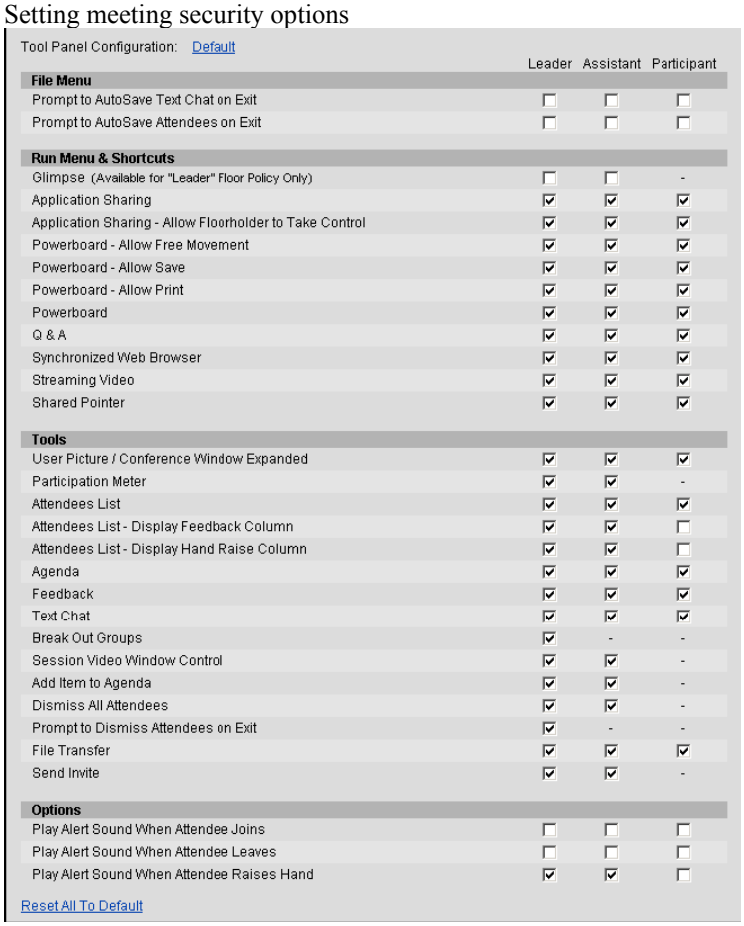
meeting scedules



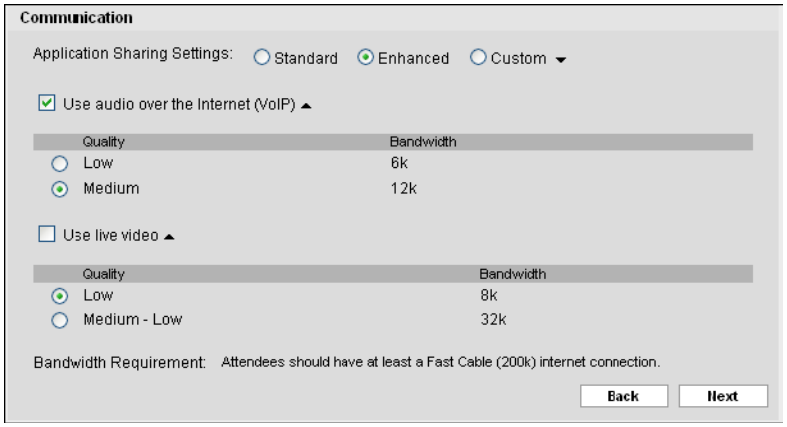
Adding content



Setting meeting security options



More security options defining options for different roles.



Communications settings

APPENDIX N: Implementation details for a new prototype of a DTC HCI

The JavaScript package with most of the list managing functionality

```
/**
 * Defines package eg.dtc_hci_functions.
 */
jsx3.lang.Package.definePackage(
               "eg.dtc_hci_functions",  // full package name
                function(dtc_hci) {  // argument in function is a short
package, good to use in this file only.

  /**
   * Returns the application server object which by default is the application
   * namespace as specified in Project->Deployment Options.
   * @returns {jsx3.app.Server} the application server object.
   * In this case: dtc1
   */
  dtc_hci.getServer = function() {
      return dtc1;
  };

  /**
   * User u_nr requests flooraccess.
   * Add a request entry for u_nr to the requestlist (rlist) and set the
request indication in the
   * userlist (ulist) for u_nr.
   */
   dtc_hci.add_request = function (u_nr) {

     // Get lists
     var ulist = dtc_hci.getServer().getJSXByName("ulist");
     var rlist = dtc_hci.getServer().getJSXByName("rlist");

     //Get user record
     var urecord = ulist.getRecord(u_nr)

     //No question record may yet exist
     if (rlist.getRecord(u_nr) == null) {

         //update user record with request
         ulist.insertRecordProperty(u_nr, "jsxfreq", "y", true)
         ulist.repaint();

         // list will grow by 1 row every time a new record is inserted
         //rlist.setGrowBy(1);

         // Get number of requests to calculate new number
         var rcount =
dtc_hci.getServer().getCache().getDocument("requestlist_xml").selectNodes("//r
ecord").getLength();

         var objRecord = new Object(); // new CDF record obj

         //Set request
         objRecord.jsxid = u_nr;
         objRecord.jsxrnr = rcount + 1;
         objRecord.jsxname = urecord.jsxname;
         objRecord.jsxqsumm =
dtc_hci.getServer().getJSXByName("rsumm_textbox").getValue();
         objRecord.jsxfstate = "read";
```

```
        objRecord.jsxfreq = "y";
        rlist.insertRecord(objRecord, null, true);
        rlist.repaint();
    }
  };

 /**
  * User u_nr withdraws a request for flooraccess.
  * Remove a request entry from u_nr from the requestlist (rlist) and reset
the request indication in the
  * userlist (ulist) for u_nr.
  */
  dtc_hci.withdraw_request = function (u_nr) {

    // Get lists
    var ulist = dtc_hci.getServer().getJSXByName("ulist");
    var rlist = dtc_hci.getServer().getJSXByName("rlist");

    //Get user record
    var urecord = ulist.getRecord(u_nr)

    //Question record must exist and floor may not be given
    if (rlist.getRecord(u_nr) != null)
      if (urecord.jsxfstate == "read") {

        //update user record with request deletion
        ulist.insertRecordProperty(u_nr, "jsxfreq", "n", true)
        ulist.repaint();

        //delete request
        rlist.deleteRecord(u_nr, true);
        rlist.repaint();
      }
  };


 /**
  * Controller denies a request for flooraccess to user u_nr.
  * Remove an entry from u_nr from the requestlist (rlist) and reset the
request indication in the
  * userlist (ulist) for u_nr.
  */
  dtc_hci.deny_request = function (u_nr) {

    // Get lists
    var ulist = dtc_hci.getServer().getJSXByName("ulist");
    var rlist = dtc_hci.getServer().getJSXByName("rlist");

    //Get user record
    var urecord = ulist.getRecord(u_nr)

    //Question record must exist and floor may not be given
    if (rlist.getRecord(u_nr) != null)
      if (urecord.jsxfstate == "read") {

        //update user record with request deletion
        ulist.insertRecordProperty(u_nr, "jsxfreq", "n", true)
        ulist.repaint();

        //delete request
        rlist.deleteRecord(u_nr, true);
        rlist.repaint();

        //set focus on userlist
        dtc_hci.focuslist ("ulist");
      }
  };
```

```
  /**
   * Controller grants flooraccess (both with and without a request) to user
u_nr.
   * Make a requestlist entry if there was none.
   * Reset the request indication on the requestlist (rlist) and the userlist
(ulist) for u_nr.
   * Set the floor reeadwrite indiaction for both the lists.
   */
   dtc_hci.grant_floor = function (u_nr) {

     // Get lists
     var ulist = dtc_hci.getServer().getJSXByName("ulist");
     var rlist = dtc_hci.getServer().getJSXByName("rlist");

     //Get user record
     var urecord = ulist.getRecord(u_nr)

     //Question record must exist, else make one.
     if (rlist.getRecord(u_nr) == null) {
        var objRecord = new Object(); // new CDF record obj

         // Get number of requests
         var rcount =
dtc_hci.getServer().getCache().getDocument("requestlist_xml").selectNodes("//r
ecord").getLength();

         //insert requestlist record
         objRecord.jsxid = u_nr;
         objRecord.jsxrnr = rcount + 1;
         objRecord.jsxname = urecord.jsxname;
         objRecord.jsxqsumm = "";
         objRecord.jsxfstate = "readwrite";
         objRecord.jsxfreq = "n";
         rlist.insertRecord(objRecord, null, true);
         rlist.repaint();
     }
     else {

         //update request
         rlist.insertRecordProperty(u_nr, "jsxfreq", "n", true)
         rlist.insertRecordProperty(u_nr, "jsxfstate", "readwrite", true)
         rlist.repaint();
     };

     //update user record with request deletion
     ulist.insertRecordProperty(u_nr, "jsxfreq", "n", true)
     ulist.insertRecordProperty(u_nr, "jsxfstate", "readwrite", true)
     ulist.repaint();

     //set focus on userlist
     dtc_hci.focuslist ("ulist");

    };


  /**
   * User u_nr returns flooraccess.
   * Reset the request indication on the requestlist (rlist) and the userlist
(ulist)
   * and set the floor reeadwrite indication for both the lists
   */
   dtc_hci.return_floor = function (u_nr) {

     // Get lists
     var ulist = dtc_hci.getServer().getJSXByName("ulist");
     var rlist = dtc_hci.getServer().getJSXByName("rlist");
```

```
     //Get user record
     var urecord = ulist.getRecord(u_nr)

     //Question record must exist and floor must be given
     if (rlist.getRecord(u_nr) != null)
       if (urecord.jsxfstate == "readwrite") {

         //update user record with request deletion
         ulist.insertRecordProperty(u_nr, "jsxfstate", "read", true)
         ulist.repaint();

         //delete request
         rlist.deleteRecord(u_nr, true);
         rlist.repaint();
       }
   };

  /**
   * Controller grabs flooraccess from user u_nr.
   * Remove an entry from u_nr from the requestlist (rlist) and set the and
set the floor reead indication
   * for the userlist (ulist) for u_nr.
   */
  dtc_hci.grab_floor = function (u_nr) {

     // Get lists
     var ulist = dtc_hci.getServer().getJSXByName("ulist");
     var rlist = dtc_hci.getServer().getJSXByName("rlist");

     //Get user record
     var urecord = ulist.getRecord(u_nr)

     //Question record must exist and floor must be given
     if (rlist.getRecord(u_nr) != null)
       if (urecord.jsxfstate == "readwrite") {

         //update user record with request deletion
         ulist.insertRecordProperty(u_nr, "jsxfstate", "read", true)
         ulist.repaint();

         //delete request
         rlist.deleteRecord(u_nr, true);
         rlist.repaint();
       };

     //set focus on userlist
     dtc_hci.focuslist ("ulist");

   };

   /**
   * Controller grabs flooraccess from all users.
   * Loop through the userlist and grab floor from each user that has a floor.
   */
    dtc_hci.grab_all_floors = function () {

     //get userlist to itereate through
     var objNodes =
dtc_hci.getServer().getCache().getDocument("userlist_xml").selectNodes("//reco
rd");

     //iterate through userlist
     for(var i=0;i<objNodes.getLength();i++) {
         //get node for the iteration
         var objNode = objNodes.getItem(i);
         //grab floor for all who have floor granted
```

```
         if (objNode.getAttribute("jsxfstate") == "readwrite") {
             dtc_hci.grab_floor (objNode.getAttribute("jsxid"));
         }
       };
    //set focus on userlist
    dtc_hci.focuslist ("ulist");

   };

   /**
   * Set the key-focus to the userlist.
   * This enables the user to move through the list with the cursorkeys
   * and activate the hotkeys for floor/request grant, grab and denial.
   */
   dtc_hci.focuslist = function (lname) {

    var lst = dtc_hci.getServer().getJSXByName(lname);
    var lst_ind = lst.getValue();
    if (lst_ind == null) {
        //no selection set: make one
        lst_ind = lst.getSortedIds()[0];
        if (lst_ind != null)
            lst.focusRowById(lst_ind);
    } else {
        lst.focusRowById(lst_ind);
    }
  };

});
```

Formathandler (in ulist) to display or hide the Grant floor icon-button

```
function(element, cdfkey, matrix, column, rownumber, server) {
        var objRecord = matrix.getRecordNode(cdfkey);
        var myText = objRecord.getAttribute("jsxfstate");
        var myText2 = objRecord.getAttribute("jsxfreq");
       if (myText2 == "y")
          element.innerHTML = '';
       if (myText == "readwrite")
          element.innerHTML = ''
};
```

Formathandler (in ulist and rlist) to display or hide the Grant request and Deny request icon-button

```
function(element, cdfkey, matrix, column, rownumber, server) {
        var objRecord = matrix.getRecordNode(cdfkey);
        var myText = objRecord.getAttribute("jsxfreq");
       if (myText == "n")
            element.innerHTML = ''

};
```

Formathandler (in ulist and rlist) to display or hide the Grab floor icon-button

```
function(element, cdfkey, matrix, column, rownumber, server) {
        var objRecord = matrix.getRecordNode(cdfkey);
        var myText = objRecord.getAttribute("jsxfstate");
       if (myText == "read")
            element.innerHTML = ''

};
```

XML Format of the user-list (both stored as file and in the local cache)

```
<data jsxid="jsxroot">
        <record jsxid="usernr" jsxname="name" jsxfstate="read" or "readwrite"
        jsxfreq="y" or "n"/>

Cached reference: userlist_xml
Is bound to: ulist
HD path for initial contents (relative): xml/userlist.xml
```

XML Format of the request-list (stored in the local cache)

```
<data jsxid="jsxroot">
      <record jsxid="usernr" jsxrnr="request nr" jsxname="name"
jsxqsumm="request summary" jsxfstate="read" or "readwrite"
        jsxfreq="y" or "n"/>
</data>

Cached reference: requestlist_xml
Is bound to: rlist
```

APPENDIX O: Listing of the DTC HCI implementation in TCL/TK

```
#!/Window/X11/contrib/bin/wish8.0
global num_images images
if {![info exists num_images]} {set num_images 0}
catch "image create photo tr_no.gif -file tr_no.gif"
set images($num_images) tr_no.gif
incr num_images
catch "image create photo tr_no.gif -file tr_no.gif"
set images($num_images) tr_no.gif
incr num_images
catch "image create photo tr_no.gif -file tr_no.gif"
set images($num_images) tr_no.gif
incr num_images
global Menu_string
global auto_path images num_images
global is_guibuilder
if {[catch {set is_guibuilder $is_guibuilder}]} {
   set is_guibuilder 0
}
if {[file exists common2.tcl]} {
  source common2.tcl
  source extensions2.tcl
  source fancylb2.tk
}
set sourcename ms_ihci.tcl
if {!$is_guibuilder} {
  catch "source ms_ihci.tcl"
  auto_mkindex . ms_ihci.tcl
}
global posfilename
set posfilename "dtc_r7.pos"
defaultpos_load $posfilename
set align_list ""
wm withdraw .
#- TOP LEVEL procedures---------------------------------

set Name .system
set Parent $Name
global Procs
set Procs($Name) {sy:kill_procs sy:reset_blockingcount
sy:pipe_write sy:write_wait sy:pipe_read sy:start_netlayer
sy:check_address sy:set_audio_out_source sy:set_audio_in_source
sy:stop_netlayer sy:emergency_stop sy:set_volume_out
sy:set_volume_in sy:set_volume_mon sy:define sy:open_db_file
sy:close_db_file sy:read_db_line sy:init_audio sy:quit_all
sy:def_images sy:get_atm_adr sy:get_eth_adr
sy:init_atm_conversion sy:get_local_adr sy:get_cmdln_params
sy:is_windows sy:audiofailure sy:get_audioprops sy:kill_afters
sy:stophci}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) sy:
set Glob_prefix($Name) sy_
proc sy:kill_procs {} {
global sy_procs_ids
global sy_windoze sy_nl_exec

#kill all procs in list procs_id and clear the list

#all that follows is not possbile in windows
if {$sy_windoze} {return}

#w_sc:addlst_debug "killing networklayer"
#No updates during this operation:
#That would crash TCL/TK !!!!

#kill known session-layer processes

catch "exec kill  $sy_procs_ids"

#empty list
set sy_procs_ids ""

#now make sure all processes are realy dead
set tpid [exec find_procs2 $sy_nl_exec]

#give opportunity to die gracefully
catch "exec kill $tpid"

#wait
after 500

#just die !
catch "exec kill -9 $tpid"

#wait
after 500

#clearing up shared-memory mess.
set memlst [string trim "[exec find_mem]"]

foreach m $memlst {
    catch "exec ipcrm -m $m"
}
}
proc sy:reset_blockingcount {} {

  global sy_blocking_count

  set sy_blocking_count 0

  after 10000 {sy:reset_blockingcount}

}
proc sy:pipe_write s {

  global sy_pipe sy_stopping sy_sl_emu

  if {$sy_sl_emu ==1} {
      w_sc:addlst_pipe_out "EMU: $s"
     return
  }

   sy:write_wait

  if {$sy_stopping == 1} {
    return
  }

  w_sc:addlst_pipe_out "$s"
  catch {puts $sy_pipe "$s"}

  if {[catch {flush $sy_pipe}]} {
    main:status_message "ERROR: networklayer stopped !!!"
    sy:emergency_stop 1 1 "Networklayer stopped !!"
  }
}
proc sy:write_wait {} {

global sy_pipewrite sy_windoze sy_stopping sy_sl_emu

if {$sy_sl_emu ==1} {return}

# if {$sy_pipewrite == 0} {
#    tkwait variable sy_pipewrite
# }

  if {$sy_pipewrite == 0} {
    while {$sy_pipewrite == 0} {
      wait 50
      if {$sy_stopping == 1} {
        return
      }
      update
    }
  } else {
    if {$sy_windoze} {
      read before writing, so pipe-buffer cannot be overrun
      sy:pipe_read 1
    }
  }
}
proc sy:pipe_read single {

  global sy_pipe sy_pipewrite sy_blocking_count sy_stopping
sy_windoze sy_sl_emu

  set sy_pipewrite 0

if {$sy_sl_emu==1} {return}

  if {$sy_stopping == 1} {
     return
  }

  if {$sy_windoze} {
    set sy_blocking_count 1
  } else {
    #is there a runaway process on the other end of the pipe ?
    set sy_blocking_count [expr $sy_blocking_count + 1]
    if {$sy_blocking_count > 50} {
      #emergency shutdown
      set sy_pipewrite 1
      sy:emergency_stop 1 1 "ERROR !!! Networklayer stopped
responding !"
      return
    }
  }

  set p_end 1

  catch {set p_end [eof $sy_pipe]}

  if {$p_end} {

    set sy_pipewrite 1
    sy:emergency_stop 1 1 "MAJOR ERROR !!! Communication broken
with networklayer"
    return

  } else {

    if {$sy_stopping == 1} {
       return
    }
    parse pipe-message
    set inp "[gets $sy_pipe]"

    if {$inp != ""} {
      w_sc:addlst_pipe_in "$sy_blocking_count $inp"
      set sy_pipewrite 1
      nl:eval_incoming $inp
```

```
      #update idletasks
    } else {
     set sy_pipewrite 1
    }
  }

  if {$sy_windoze && !$single} {
    #setup file-event handler for bound pipe again
    after 100 {sy:pipe_read 0}
  }
}
proc sy:start_netlayer {user_id refl_adr atm_adr sl_emu} {
#Start video process(es) and keep track of
#their process-id's

#video-settings and process-ids are global
global sy_pipe sy_stopping sy_pipewrite sy_procs_ids
global sy_old_user_id sy_old_refl_adr sy_old_atm_adr
global sy_windoze sy_nl_exec sy_sl_emu

#we are NOT stopping
set sy_stopping 0

#store number for emergency recovery
set sy_old_user_id $user_id
set sy_old_refl_adr $refl_adr
set sy_old_atm_adr $atm_adr
set sy_sl_emu $sl_emu

#make sure all video processes are dead
#before starting one. (list is cleared)

if {$sy_sl_emu ==1} {
   w_sle:show
   return
}

sy:kill_procs
after 100

#start audiocontrol, keep process-id
#set sy_procs_ids [exec audiocontrol &]

#start blocking detection utility
if {! $sy_windoze} {sy:reset_blockingcount}

#Start video process and store file-id of the pipe to the
#main process in sy_pipe.

#check for atm address
if {$atm_adr != ""} {
  set temp "-A $atm_adr"
} else {
  set temp ""
}

#user id can't be longer than 18 characters
set user_id [string range $user_id 0 17]

w_sc:addlst_debug "Starting networklayer $sy_nl_exec"

#Start network layer and bind a pipe to it.

if {$sy_windoze} {
  set sy_pipe [open "| $sy_nl_exec -R $refl_adr -N $user_id
$temp" "r+"]
} else {
  #use a seperate shell to set the right environment
  set sy_pipe [open "| tcall $sy_nl_exec -R $refl_adr -N
$user_id $temp" "r+"]

}

#get process_id's of processes on other end of pipe.
append sy_procs_ids " [pid $sy_pipe]"

set sy_pipewrite 0

#Setup file-event handler for bound pipe.
if {$sy_windoze} {
  fconfigure $sy_pipe -blocking 0
  after 150 {sy:pipe_read 0}
} else {
  fileevent $sy_pipe readable {sy:pipe_read 0}
}

set sy_pipewrite 1

update idletasks
}
proc sy:check_address {refl_adr atm_adr sl_emu} {

#Check reflectoradress

  global sy_windoze

  if {$sl_emu ==1} {return 1}

  main:status_message "Testing address.."

  #reflector ?
  if {[string length "[string trim $refl_adr]"] == 0} {

    #message
    main:status_message "NEED A REFLECTOR ADDRES !!!"
    return 0

  } else {

    #no checking in windows
    if {$sy_windoze} {return 1}

    if {$atm_adr != "" && [catch "exec ping $atm_adr 2"]} {
      #message
      main:status_message "REFLECTOR ADRESS NOT FOUND !!!"
      return 0
    } else {
      if {$atm_adr != ""} {
        #message
        main:status_message "ATM ADRESS NOT FOUND !!!"
        return 0
      }
    }
  }
  return 1
}
proc sy:set_audio_out_source {speaker head aux} {

global sy_audiofailure sy_bit_speaker sy_bit_head sy_bit_aux
global sy_windoze

if {$sy_windoze} {return}

if {$sy_audiofailure} {return}
  set outports 0
  if {$speaker} {set outports [expr $outports +
$sy_bit_speaker]}
  if {$head} {set outports [expr $outports + $sy_bit_head]}
  if {$aux} {set outports [expr $outports + $sy_bit_aux]}
  exec port_out $outports

}
proc sy:set_audio_in_source in_source {

global sy_audiofailure
global sy_windoze sy_sl_emu

if {$sy_windoze} {return}
if {$sy_audiofailure} {return}

exec port_in $in_source

}
proc sy:stop_netlayer {} {

  global  sy_pipe sy_stopped sy_killed sy_stopping
  global sy_windoze sy_sl_emu

  #stop the networklayer.

if {$sy_sl_emu ==1} {
   w_sle:hide
   return
}


  #already stopping ? then continue
  if {$sy_stopping} {
     return
   }
  set sy_stopping 1

  #destroy pipe-handler
  if {$sy_windoze == 0} {
    fileevent $sy_pipe readable {}
  }

  #stop all processes (just to be sure)
  sy:kill_procs

  catch {close $sy_pipe}

  #all is stopped
}
proc sy:emergency_stop {m recover message} {

    global sy_stopping sy_old_user_id sy_old_refl_adr
sy_old_atm_adr sy_sl_emu

    #an error has occurred: stop networklayer !
    #if recover = 1 then try to recover
    #if m=1 and recover = 1, than give message and offer
    #possibillity to recover.

    if {$sy_sl_emu ==1} {return}

    #don't stop more than once
    if {$sy_stopping} {return}

    w_sc:addlst_debug "Emergency-stop!"

    sy:stop_netlayer

    #kill all remaining video stuff
    nl:clear_video_waits

    w_sc:addlst_debug "Network stopped"

    #message window
    if {$recover} {

      #offer recovery
      if {$m} {
      set wmessage "$message There might be a possibility to
recover"
    set choice [tk_dialog ".tkdialog" "Floor Control"
"$wmessage" "" 0 "Recover" "STOP"]
      } else {
        set choice 0
      }
    } else {

      #just kill all
      set wmessage "$message"
```

```
      set choice 1
      tk_dialog ".tkdialog" "Floor Control" "$wmessage" "" 0
"STOP"
    }

    update idletasks

    if {$choice == 0} {

      #try to recover
      after 500
      sy:start_netlayer $sy_old_user_id $sy_old_refl_adr
$sy_old_atm_adr
      lm:recover_all
    } else {

     #just stop
     w_fc:stop_hci
    }
}
proc sy:set_volume_out vol {

global sy_windoze is_guibuilder

if {$is_guibuilder} {return 0}
if {$sy_windoze} {return 0}


  exec volume_out $vol

}
proc sy:set_volume_in vol {

global sy_windoze is_guibuilder

if {$is_guibuilder} {return 0}
if {$sy_windoze} {return 0}

  exec volume_in $vol

}
proc sy:set_volume_mon vol {

global sy_windoze is_guibuilder

if {$is_guibuilder} {return 0}
if {$sy_windoze} {return 0}

  exec volume_mon $vol

}
proc sy:define ex {

global sy_bit_speaker sy_bit_head sy_bit_aux
global sy_procs_ids ref_pid
global sy_stopped
global sy_pipewrite
global sy_windoze tcl_platform sy_nl_exec

#define systemlayer.
#ex is user-defined session-layer executable

#determine platform
#and set execubale for networklayer
if {$tcl_platform(platform) == "unix"} {
  set sy_windoze 0
  set sy_nl_exec "tel"
} else {
  set sy_windoze 1
  set sy_nl_exec "TelSingle.exe"
}

#if executable is given as parameter, use it.
if {$ex != ""} {
  set sy_nl_exec $ex
}

#control bits of audio device for playing sounds
set sy_bit_speaker 1
set sy_bit_head 2
set sy_bit_aux 4

#clear list of proc-id's
set sy_procs_ids ""
set ref_pid ""

set sy_stopped 1
set sy_pipewrite 1

sy:init_atm_conversion

}
proc sy:open_db_file name {

  #Open database file "name" and return file-handler

  set f_id [open "$name" "RDONLY"]
  return $f_id
}
proc sy:close_db_file f_id {

  close $f_id

}
proc sy:read_db_line f_id {

  #read one line of database

  set ln ""
  while {$ln == ""} {
    if {[eof $f_id]} {
      set ln "END."
    } else {
      set ln [string trim [gets $f_id]]
      #skip remarks
      if {[string index $ln 0] == "\#"} {
        set ln ""
      }
    }
  }
  return $ln
}
proc sy:init_audio {} {

  #get volume
  global out_vol_bar in_vol_bar sy_stopped sy_audiofailure

  global sy_windoze

  set sy_stopped 0
  if {$sy_windoze} {
    set sy_audiofailure 0
  } else {
    #check audio and
    #start reading audio-settings
    set sy_audiofailure 0
    set outvol [sy:get_audioprops]
    if {$outvol < 0} {
      main:status_message "Audio Device not working !!!"
      set sy_audiofailure 1
    }
  }
  return $sy_audiofailure
}
proc sy:quit_all {} {

catch {sy:kill_procs}

#quit all windows
exit
}
proc sy:def_images {} {

  #images
  set pal "3/3/2"
  image create photo stop -file "stop.gif" -format gif89 -
palette $pal
  image create photo not_allowed -file "tr_no.gif" -format
gif89 -palette $pal
  image create photo light_green -file "trl_green.gif" -format
gif89 -palette $pal
  image create photo light_yellow -file "trl_yellow.gif" -
format gif89 -palette $pal
  image create photo light_red -file "trl_red.gif" -format
gif89 -palette $pal
  image create photo light_green2 -file "trl_green2.gif" -
format gif89 -palette $pal
  image create photo light_yellow2 -file "trl_yellow2.gif" -
format gif89 -palette $pal
  image create photo light_red2 -file "trl_red2.gif" -format
gif89 -palette $pal

}
proc sy:get_atm_adr e_adr {

  global sy_eth_atm

  set adr $e_adr
  catch {set adr $sy_eth_atm($e_adr)}

  return $adr
}
proc sy:get_eth_adr adr {

  global sy_eth_atm

  set e_adr $adr
  foreach a [array names sy_eth_atm] {
    if {$adr == $sy_eth_atm($a)} {
      set e_adr $a
      break
    }
  }
  return $e_adr
}
proc sy:init_atm_conversion {} {

  global sy_eth_atm

  set sy_eth_atm(banach) "atmli006"
  set sy_eth_atm(boole) "atmli004"
  set sy_eth_atm(ferrari) "atmli005"

}
proc sy:get_local_adr {} {

  global sy_windoze

  #this function is not yet defined when ATM is used in windows
  if {$sy_windoze} {return ""}

  #return networkname of local machine
  set adr [exec uname -n]

  return $adr
}
proc sy:get_cmdln_params {} {

  global is_guibuilder argv

  if {$is_guibuilder} {
    return ""
  } else {
    return $argv
```

```
  }
}
proc sy:is_windows {} {

  global sy_windoze
  return $sy_windoze

}
proc sy:audiofailure {} {

  #return audiofailure. This is TRUE when there is
  #a problem with the audio-device

  global sy_audiofailure

  return $sy_audiofailure
}
proc sy:get_audioprops {} {

global sy_bit_speaker sy_bit_head sy_bit_aux
global sy_windoze sy_audiofailure

#Get all audio volume- and port-values
#and pass them to them HCI.
#repeat this every 5 seconds.

#This is not yet possible for Windows.
#this procedure is only for SUN's

#retreive input, output and monitor volume and set the
#Pass values to HCI.
#check also for audio-problems

catch {
  set outvol [exec volume_out]
  lm:ai:show_volume $outvol
  lm:ao:show_volumes [exec volume_in] [exec volume_mon]
}

#now retriev all ports
set port_out [exec port_out]
set out_speaker [expr ($port_out & $sy_bit_speaker) > 0]
set out_phone [expr ($port_out & $sy_bit_head) > 0]
set out_aux [expr ($port_out & $sy_bit_aux) > 0]

#Pass values to HCI

catch {
  lm:ai:show_source $out_speaker $out_phone $out_aux
  lm:ao:show_source [exec port_in]
}

update idletasks

if {$outvol >= 0} {
  #audiodevice works
  #do this again in about 4 seconds
  after 5000 [list sy:get_audioprops]
}

return $outvol
}
proc sy:kill_afters {} {

  #kill all timed procedures

  #find the ID's
  set ids [after info]

  #cancel all
  foreach id $ids {
    after cancel $id
  }
}
proc sy:stophci {} {

  global is_guibuilder

  #no HCI-activity when guibuilder is active

  return $is_guibuilder

}
#- TOP LEVEL procedures---------------------------------

set Name .network_layer
set Parent $Name
global Procs
set Procs($Name) { nl:grant_floor nl:grab_floor nl:grasp_floor
nl:refuse_floor nl:broadcast_textcontrol nl:signal_video_ready
nl:clear_video_waits nl:stop_netlayer nl:eval_incoming
nl:return_floor nl:withdraw_request nl:video_wait_write
nl:init_vid_coding nl:m_compress nl:m_expand
nl:enable_vid_write nl:disable_vid_write nl:enable_au_read
nl:enable_au_write nl:disable_au_write nl:broadcast_text
nl:set_mixing_volume nl:request_floor nl:start_network
nl:enable_reception nl:prepare_vid_out_props
nl:disable_video_read nl:enable_video_read
nl:disable_all_video_read nl:enable_all_video_read
nl:add_video_source nl:delete_video_source nl:set_vidr_auto
nl:disable_au_read}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) nl:
set Glob_prefix($Name) nl_
proc nl:grant_floor {media ipnr} {

  #Grant floor (read and/or write) by giving a token

  sy:pipe_write "Tokengive_req [nl:m_compress $media] $ipnr"


}
proc nl:grab_floor {media ipnr} {

   #Grab floor from participant (read and/or write)
   #by grabbing his token.

   sy:pipe_write "FC_Tokengrab_req [nl:m_compress $media]
$ipnr"

}
proc nl:grasp_floor media {

  #Grasp floors in "media" from all participants

  sy:pipe_write "FC_Reset_req [nl:m_compress $media]"

}
proc nl:refuse_floor {media ipnr userdata} {

   #rufes floor request for "media" to the user at "ipnr".
   #"userdata" may contain a message to that user.

   #add extra space because the networkprogram
   #crashes when "userdata is empty.
   set userdata " $userdata"

   sy:pipe_write "FC_Reject_req [nl:m_compress $media] $ipnr
$userdata"

}
proc nl:broadcast_textcontrol c_text {

  #broadcaste text on textchannel 0.

  #this is not an ordinary channel and will not be
  #displayed as text.It can be used for auxilary protocols
  #that are not yet implemented within the network program.

  catch {sy:pipe_write "text_stream_req 0 $c_text"}

}
proc nl:signal_video_ready {direction operation} {

  #This procedure must be called when a signal has been
  #received that the current video operation has been
  #completed.

  global nl_video_busy nl_video_commandlist nl_vid_write_error
nl_vid_read_error

  w_sc:addlst_debug "Videocommand $direction $operation ready."

  if {$direction == "in"} {
    if {$operation == "initialized"} {
     # video sending has started and everything is ok.
       set nl_vid_write_error 0
    }
    if {$operation == "stopped"} {
    # video sending has stopped
      }
    if {$operation == "error"} {
      #It is not possible to transmit video
      #Mark this for the future.
      set nl_vid_write_error 1
      #Define floors accordingly and give message
      fu:video_write_unusable
      #restart the networklayer to prevent further errors
      sy:emergency_stop 0 1 ""
    }
  } else {
    if {$operation == "error"} {
      #It is not possible to receive video
      #Mark this for the future.
      set nl_vid_read_error 1
      #Define floors accordingly and give message
      fu:video_read_unusable
      #restart the networklayer to prevent further errors
      sy:emergency_stop 0 1 ""
    }
  }

  #If there are there other videocommands in the que,
  #execute the next one

  if {[llength $nl_video_commandlist] > 0} {
    #get first command in list
    set command [lindex $nl_video_commandlist 0]
    #delete it from list
    set nl_video_commandlist [lreplace $nl_video_commandlist 0
0]
    #execute it
    sy:pipe_write $command
  } else {
    #no more commands in list
    set nl_video_busy 0
  }
}
proc nl:clear_video_waits {} {

  #clear all blocked video commands
  #and set all video-in as disabled

  global nl_video_commandlist nl_video_busy
  global nl_vidr_enabled nl_all_vidr_enabled

  set nl_all_vidr_enabled 0
  foreach ipnr [array names nl_vidr_enabled] {
    set nl_vidr_enabled($ipnr) 0
  }
  w_sc:addlst_debug "video receivelist cleared"

  set nl_video_commandlist ""
  set nl_video_busy 0
```

```
  w_sc:addlst_debug "video commandlist cleared"

}
proc nl:stop_netlayer {} {

  #stop the whole networklayer.
  #This means that both the TCL-networklayer and the
  #session-layer execuable must be stopped.

  #give message
  w_sc:addlst_debug "Stopping transfer.."
  update idletasks

  #stop session-layer
  catch {sy:pipe_write "quit"}

  #wait for processes to die "naturally"
  after 2000

  #make sure the networklayer stops
  sy:stop_netlayer

  #Stop video transfer
  nl:clear_video_waits

  main:status_message "Networklayer stopped.."

}
proc nl:eval_incoming inp {

global nl_reception nl_in_store
global nl_vidr_auto_add

#evaluate incoming pipe-messages. These are send by
#the actual session-layer program.
#For each message the parameters must be extracted and,
#if neccesary, converted.then the corrseponding
#procedure(s) must be called.

#check if interpretation is allowed.
if {$nl_reception} {

 #interpret message
 switch -regexp -- $inp {

     Video_status* {

       #video is running
       set direction [lindex $inp 1]
       set operation [lindex $inp 2]
       nl:signal_video_ready $direction $operation
    }

     adding* {

       #someone joins the conference
       set ipnr [lindex $inp 1]
       set name "[lindex $inp 2]"
       #prepare video reception when neccesary
       if {$nl_vidr_auto_add} {nl:add_video_source $ipnr}
       fc:add_floor_user $ipnr "$name"
     }

     deleting* {

       #someone leaves the conference
       set ipnr [lindex $inp 1]
       nl:delete_video_source $ipnr 0
       fc:delete_floor_user $ipnr
     }

     FP_Tokenplease_ind* {

      #someone wants to speak
        set ipnr [lindex $inp 2]
        set media [nl:m_expand [lindex $inp 1]]
        set userdata [lrange $inp 3 end]
        fc:floor_requested $media $ipnr $userdata
    }

    FC_Reject_ind* {

      #teacher rejects tokenplease
        set ipnr [lindex $inp 2]
        set media [nl:m_expand [lindex $inp 1]]
        set userdata [lrange $inp 3 end]
        fc:floor_refused $media $ipnr $userdata
     }

    FP_Withdraw_ind* {

      #someone cancels a request to speak
      set ipnr [lindex $inp 2]
      set media [nl:m_expand [lindex $inp 1]]
      fc:floor_canceled $media $ipnr
    }

    Tokengive_ind* {

        set ipnr [lindex $inp 2]
        set media [nl:m_expand [lindex $inp 1]]

        fc:floor_given $media $ipnr
    }

    FC_Tokengrab_conf* {

        set ipnr [lindex $inp 2]
        set media [nl:m_expand [lindex $inp 1]]

        fc:floor_was_grabbed $media $ipnr
    }

    FC_Tokengrab_ind* {

      set media [nl:m_expand [lindex $inp 1]]
      fc:floor_grabbed $media
    }

    FC_Reset_ind* {

      set media [nl:m_expand [lindex $inp 1]]
      set ipnr [lindex $inp 2]
      fc:floor_grasped $media $ipnr
    }

    text_stream_ind* {

      set channel [lindex $inp 1]
      if {$channel == 0} {

        #control channel

        #This will be used for service primitives that
        #are (not yet) implemented in the underlying
        #network-layer

        switch -regexp -- [lindex $inp 3] {

          FC_announce* {
            #Floor controller announced:
            #Keep address for later use.

            #set media [nl:m_expand [lindex $inp 4]]
            #set c_ipnr [lindex $inp 2]
            #fc:floor_controlled $media $c_ipnr
          }
        }
      } else {

        #Normal text

        fu:text_in $channel "[lindex $inp 2]" "[lreplace $inp 0
2]"
      }
     }

     "bus error*" {
       sy:emergency_stop 1 1 "BUS ERROR !!! Networklayer
crashed !"
     }

     "segmentation fault*" {
       sy:emergency_stop 1 1 "SEGMENTATION FAULT !!!
Networklayer crashed !"
     }


     Error* {

       #something's rotten in the state of Denmark !
       main:status_message $inp
     }
   }
 } else {
   #interpretation not yet allowed.
   #store messages without interpretation.
   lappend ln_in_store $inp
 }
}
proc nl:return_floor {media ipnr} {

  #return "media" to floor floor controller at "ipnr".

  sy:pipe_write "Tokengive_req [nl:m_compress $media] $ipnr"

}
proc nl:withdraw_request {media ipnr} {

  #withdraw a request to the controller at "ipnr"
  #for "media"

  sy:pipe_write "FP_Withdraw_req [nl:m_compress $media] $ipnr"

}
proc nl:video_wait_write command {

  #This procedure manages videocommands to prevent
  #the ecetution of more the one command at a time
  #by the session layer (may cuase a crash).

  #whenever a video-command is being executed,
  #the rest is collected in a waiting-list.
  #These will be executed one at a time.

  global nl_video_commandlist nl_video_busy

  if {$nl_video_busy} {
    #put videocommand in the execution-list
    lappend nl_video_commandlist "$command"
    w_sc:addlst_debug "$command added to videocommandlist"
  } else {
    #execute videocommand directly
    set nl_video_busy 1
    sy:pipe_write $command
  }


}
proc nl:init_vid_coding vcod {

#Initialise all variables for video transmission and
#-reception.
#Set the video-coding "vcod"(Mpeg1, CellB, Jpeg or YUVY)
```

```
#for both reception and transmission.
#clear/initialise the video-command queu

global nl_video_w_compr nl_video_r_compr
global nl_video_w_starting nl_video_w_enabled
global nl_video_busy nl_vid_write_error
global nl_video_w_stopping nl_vid_read_error
global nl_vidr_enabled nl_all_vidr_enabled nl_vidr_auto_add
global nl_vsize nl_vpreview

set nl_video_w_compr $vcod
set nl_video_r_compr $vcod
set nl_video_w_enabled 0
set nl_video_w_starting 0
set nl_video_w_stopping 0
set nl_video_busy 0
set nl_vid_write_error 0
set nl_vid_read_error 0

set nl_vidr_auto_add 0
set nl_all_vidr_enabled 0
catch {unset nl_vidr_enabled}

#default parameters for video-out
set nl_vsize 3
set nl_vpreview 1

nl:clear_video_waits
}
proc nl:m_compress media {

  #compress media so there are no spaces anymore in between
  #the arguments.

  set new_m ""

  foreach m $media {
    if {[string length $m] < 3} {
      #when a parameter contains no direction,
      #assume "write".
      set m "W$m"
    }
    set new_m "$new_m$m"
  }
  return $new_m
}
proc nl:m_expand media {


  #expands media so there are spaces in between
  #the arguments.
  #this means that this string can be treated
  #like a TCL-list again.

  set new_m ""

  for {set t 0} {$t < ([string length "$media"] - 2 )} {incr t
3} {
    lappend new_m [string range $media $t [expr $t + 2]]
  }
  return $new_m
}
proc nl:enable_vid_write {} {

  #set the right parameters for video-out and
  #start the actual video-transfer,
  #if video-transfer was not yet enabled.

  #The actual capture and transmission of video is done
  #by the sessionlayer.

  #Checking for transfer already being enabled is
  #necessary to prevent the sessionlayer from crashing

  global nl_video_w_compr nl_vpreview nl_vsize nl_video_busy
nl_video_w_enabled nl_vid_write_error

  if {($nl_video_w_enabled == 0) && ($nl_vid_write_error == 0)}
{
    set nl_video_w_enabled 1
    #settings
    w_sc:addlst_debug "video write starting"
    nl:video_wait_write "set_video_in_encoding
$nl_video_w_compr"

    nl:video_wait_write "set_video_in_scale $nl_vsize"

    if {$nl_vpreview} {
      nl:video_wait_write "set_video_in_preview_on"
    } else {
      nl:video_wait_write "set_video_in_preview_off"
    }

    #start transfer
    nl:video_wait_write "video_stream_enabled"
  }
}
proc nl:disable_vid_write {} {

  #Stop video-transfer if video transfer was enabled.

  #Checking for transfer already being disabled is
  #necessary to prevent the sessionlayer from crashing

  global nl_video_w_enabled nl_vid_write_error

  #stop video write to network: blocking procedure

  if {($nl_video_w_enabled == 1) && ($nl_vid_write_error == 0)}
{
      set nl_video_w_enabled 0
      w_sc:addlst_debug "video write stopping"
      nl:video_wait_write "video_stream_disabled"
  }
}
proc nl:enable_au_read {} {

  #enable audio-reception, if possible.

  #The actual reception and playback of audio is done by the
  #sessionlayer.

  if {![sy:audiofailure]} {
    after 200
    sy:pipe_write "enable_read_stream"
  }
}
proc nl:enable_au_write {} {

  #enable audio transmission, when possible.

  #The actual recording and transmission of audio is done
  #by the sessionlayer.

  if {![sy:audiofailure]} {
    after 200
    sy:pipe_write "audio_stream_enabled"
  }
}
proc nl:disable_au_write {} {

  #disable audio transmission, when needed.

  if {![sy:audiofailure]} {
    after 200
    sy:pipe_write "audio_stream_disabled"
  }
}
proc nl:broadcast_text {ch text} {

  #broadcast a tekst (max 255 charcters) to all
  #participants on textchannel "ch".

  #limit text to 255 characters
  set text [string range $text 1 255]

  sy:pipe_write "text_stream_req $ch $text"
}
proc nl:set_mixing_volume {ipnr value} {

  #set relative mixing-volume for audio reception from a
  #single participant
  #To disable audio reception from a single participant,
  #set this to 0.

  sy:pipe_write "set_stream_volume $ipnr $value"

}
proc nl:request_floor {media ipnr userdata} {

  #Request an access token for the floor(s) in "media" to
  #the floor controller at "ipnr".
  #Extra information about the request can be given in
  #"userdata".

  #add extra space
  #(else the other end of the pipe collapses)
  set userdata " $userdata"

  sy:pipe_write "FP_Tokenplease_req [nl:m_compress $media]
$ipnr $userdata"

}
proc nl:start_network {n refl a_atm sl_emu} {

  #start the actual session-layer and prevent
  #incoming network events from reaching the
  #upper layers before they are properly initialised.

  #"n" is the name or number from the participant.
  #For now, this must alwayse be a name.
  #"refl" is the reflector-address.
  #a_atm is the IP-ATM-address of this machine. This is
  #only necessary for an IP-over-ATM connection.

  global nl_reception nl_in_store

  #Do not yet pass messages through
  set nl_reception 0
  set nl_in_store ""

  #give message
   main:status_message "Starting networklayer"

   sy:start_netlayer $n $refl $a_atm $sl_emu
}
proc nl:enable_reception {} {

  #This must be called when all upper layers are
  #properly initialised. It passes alle pending
  #network events to the parser.

  global nl_reception nl_in_store

  #Now stored incoming messages can be interpreted

  #give message
  main:status_message "Intialisation complete."

  set nl_reception 1
  foreach i $nl_in_store {
    nl:eval_incoming $i
  }
```

```
  set in_store ""
}
proc nl:prepare_vid_out_props {v p} {

  #set extra parameters for video transfer
  #and check if size are possible
  #return the actual accepted size

  #v = video size (range 0 to 4, with 0 as small)
  #p = preview enabled (1) or disabled (0)

  #The size determines the actual transfer size as well
  #as the size of the optional preview-window

  global nl_vsize nl_vpreview nl_video_w_compr

  #size-parameter ranges from 0 to 4. 0 is large, 4 is small
  #conversion takes place here !
  set nl_vsize [expr 4 - $v]

  set nl_vpreview $p

  #Mpeg1 only works for size = 2
  if {$nl_video_w_compr == "Mpeg1"} {
    set nl_vsize 2
    set v 2
  #  lm:vo:set_vsizebar [expr 4 - $vsize]
  }

  #CellB cannot be larger than size = 2 (size 0 = largest)
  if {$nl_video_w_compr == "CellB"} {
    if {$nl_vsize < 2} {
      set nl_vsize 2
      set v 2
      #show to user
      #lm:vo:set_vsizebar [expr 4 - $vsize]
    }
  }

#return actual video-size
return "$v"
}
proc nl:disable_video_read ipnr {

  #disable video-reception for "ipnr", if necessary.

  global nl_vidr_enabled

  #For all registered video-transmitters, the state of
  #video-recpetion (enabled/disabled) is kept in one array.
  catch {
    if {$nl_vidr_enabled($ipnr) != 0} {
       w_sc:addlst_debug "video read stopping for $ipnr"
       set nl_vidr_enabled($ipnr) 0
       nl:video_wait_write "hide_video_stream $ipnr"
    }
  }
}
proc nl:enable_video_read ipnr {

  #enable video-reception for "ipnr", if nessecary
  #and possible.

  #This can be done, even if that participant
  #is currently not (or never) transmitting video.
  #Video-reception will then be pending and will be
  #activated the instant that participant starts
  #transmitting.
  #pending video-receptions can be disabled safely.

  #The actual reception and playback of video is done
  #by the sessionlayer.

  #Checking for reception already being enabled is
  #necessary to prevent the sessionlayer from crashing

  global nl_vidr_enabled nl_video_r_compr nl_vid_read_error

  if {$nl_vid_read_error} {return}

  #For all registered video-transmitters, the state of
  #video-recpetion (enabled/disabled) is kept in one array.
  catch {
    if {$nl_vidr_enabled($ipnr) == 0} {
      w_sc:addlst_debug "video read starting for $ipnr with
$nl_video_r_compr"
      set nl_vidr_enabled($ipnr) 1
      nl:video_wait_write "set_video_out_decoding $ipnr
$nl_video_r_compr"
      nl:video_wait_write "display_video_stream $ipnr"
    }
  }
}
proc nl:disable_all_video_read {} {

  #Disable video-reception for all registered
  #video-transmitting participants, if nessecary.

  global nl_vidr_enabled nl_all_vidr_enabled

  set nl_all_vidr_enabled 0
  foreach ipnr [array names nl_vidr_enabled] {
    nl:disable_video_read $ipnr
  }
}
proc nl:enable_all_video_read {} {

  #Enable video-reception for all registered
  #video-transmitting participants, if nessecary.

  global nl_vidr_enabled nl_all_vidr_enabled

  set nl_all_vidr_enabled 1
  foreach ipnr [array names nl_vidr_enabled] {
    nl:enable_video_read $ipnr
  }
}
proc nl:add_video_source ipnr {

  #Register a video-transmitting participant's ipnr for
  #video-reception. If video-recpetion is enebled for all,
  #then enable this one too.

  global nl_vidr_enabled nl_all_vidr_enabled

  #video must be in disabled state
  catch {
    if {$nl_vidr_enabled($ipnr) != 0} {return}
  }

  set nl_vidr_enabled($ipnr) 0
  if {$nl_all_vidr_enabled} {
    nl:enable_video_read $ipnr
  }
}
proc nl:delete_video_source {ipnr real} {

  #Unregister a video-transmitting participant's ipnr for
  #video-reception. If video-recpetion was enebled for all,
  #then disable this one if "real" is true.

  global nl_vidr_enabled

  #return 1 of ipnr was in receptionlist.
  set result [expr ! [catch {
    if {$real} {
      nl:disable_video_read $ipnr
    }
    unset nl_vidr_enabled($ipnr)
  }]]

  return $result
}
proc nl:set_vidr_auto vi_auto {

  #If "vi_auto" is 1, then, for all added particpants,
  #they are automatically registered for video-reception.

  #This can be safely set, even if not all particpants have
  #cameras.

  global nl_vidr_auto_add

  set nl_vidr_auto_add $vi_auto
}
proc nl:disable_au_read {} {

 #disable audio-reception, if necessary.

  if {![sy:audiofailure]} {
    after 200
    sy:pipe_write "disable_read_stream"
  }
}
#- TOP LEVEL procedures---------------------------------

set Name .users_db
set Parent $Name
global Procs
set Procs($Name) {ud:name_in_meetinglist ud:add_to_meetinglist
ud:delete_from_meetinglist ud:is_in_meetinglist
ud:clear_meetinglist ud:read_user_database ud:get_username
ud:use_no_userdatabase ud:get_usernumber
ud:set_current_username ud:get_current_username }
global Proc_prefix Glob_prefix
set Proc_prefix($Name) users_db:
set Glob_prefix($Name) users_db_
proc ud:name_in_meetinglist {ipnr givenr} {

  global ud_iplist ud_namelist ud_nrlist use_numbers u_names
  #Description:
  #find the name of a member of the groupmeeting,
  #given its ip-number.
  #Return a name, accompanied by a number,
  #if it is possible and "givenr" is TRUE.

  set lnr [lsearch -exact $ud_iplist $ipnr]
  if {$lnr >= 0} {
    set result [lindex $ud_namelist $lnr]
    set nr [lindex $ud_nrlist $lnr]
  } else {
    return ""
  }

  if {$givenr} {
    if {$use_numbers && ($nr >= 0)} {
      set result "$result \($nr\)"
      return $result
    } else {
      set result "$result \(\?\)"
      return $result
    }
  } else {
    return $result
  }
}
proc ud:add_to_meetinglist {ipnr name nr} {

  global ud_iplist ud_namelist ud_nrlist

  #Add the new meeting member to all lists

  lappend ud_iplist $ipnr
  lappend ud_namelist $name
```

```
  lappend ud_nrlist $nr

}
proc ud:delete_from_meetinglist ipnr {

  global ud_iplist ud_namelist ud_nrlist

  #delete from main lists

  set lnr [lsearch -glob $ud_iplist $ipnr]
  if {$lnr >= 0 } {
    set ud_iplist [lreplace $ud_iplist $lnr $lnr]
    set ud_namelist [lreplace $ud_namelist $lnr $lnr]
    set ud_nrlist [lreplace $ud_nrlist $lnr $lnr]
  } else {
    #voorlopig geen error
  }
}
proc ud:is_in_meetinglist ipnr {

  global ud_iplist

  #is this ip-number still in ud_iplist ?

  set lnr [lsearch -exact $ud_iplist $ipnr]
  return "[expr $lnr >= 0]"
}
proc ud:clear_meetinglist {} {

  global ud_iplist ud_namelist ud_nrlist

  w_fc:plist:clear
  set ud_iplist ""
  set ud_namelist ""
  set ud_nrlist ""
}
proc ud:read_user_database {} {

  global use_numbers u_names

  #read database of names.
  #Each row consists of a user-number, followed by a name

  set use_numbers 1

  set end_db 0
  catch {unset u_names}

  set f [sy:open_db_file "tele_u.ini"]

  while {!$end_db} {
    set ln [sy:read_db_line $f]
    if {$ln == "END."} {
      set end_db 1
    } else {
      set u_names([string tolower [lindex $ln 0]]) [lindex $ln
1]
    }
  }

  sy:close_db_file $f
}
proc ud:get_username {n remark} {

  global use_numbers u_names

  #Return username from database when appropriate.
  #When n is not a number, just return "n".

  if {$use_numbers && ![catch {set dummy [expr $n + 1]}]} {
    if {$remark} {
      return "$u_names($n) \($n\)"
    } else {
      return $u_names($n)
    }
  } else {
    return $n
  }
}
proc ud:use_no_userdatabase {} {

  global use_numbers

  set use_numbers 0

}
proc ud:get_usernumber inp {

  global u_names use_numbers

  #convert to a number, if nessecary, or check it's existence

  set result ""

  if {$use_numbers} {

    if {[catch {set dummy [expr $inp + 1]}]} {
      #convert name into number

      #check all numbers for name in "inp"
      set found 0
      foreach nr [array names u_names] {
        if {[string tolower $u_names($nr)] == [string tolower
$inp]} {
          set found 1
          break
        }
      }
      if {$found} {
        set result $nr
      } else {
        set result -1
      }

    } else {
      #was already a number: check it.
      #and set result, when found
      if {[catch {set dummy $u_names($inp)}]} {
        #not an existing number
        set result -1
      } else {
        set result $inp
      }
    }
  } else {
    set result -1
  }

  return $result
}
proc ud:set_current_username {name nr} {

  global ud_current_name ud_current_usernr

  set ud_current_name $name
  set ud_current_usernr $nr

}
proc ud:get_current_username remark {

 global ud_current_name

  global use_numbers ud_current_name ud_current_usernr

  #Return current username
  #When current usernr is negative, just return the name.


  if {$remark} {
    if {$use_numbers && $ud_current_usernr >= 0} {
      return "$ud_current_name \($ud_current_usernr\)"
    } else {
      return "$ud_current_name \(\?\)"
    }
  } else {
    return $ud_current_name
  }
}
#- TOP LEVEL procedures---------------------------------

set Name .floor_req_grant_db
set Parent $Name
global Procs
set Procs($Name) {rg:add_to_reqlist rg:delete_from_reqlist
rg:clear_reqlist rg:requested_media rg:add_to_grantlist
rg:delete_from_grantlist rg:is_in_grantlist rg:clear_grantlist
rg:granted_media rg:get_whole_grantlist rg:get_requested_floors
rg:set_requested_floors rg:set_granted_floors
rg:get_granted_floors rg:add_requested_floors
rg:add_granted_floors rg:clear_granted_floors
rg:clear_requested_floors rg:req_or_all_media
rg:get_ud_request}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) rg:
set Glob_prefix($Name) rg_
proc rg:add_to_reqlist {media ipnr userdata} {

global rg_req_iplist rg_req_media_list rg_req_ud_list

  #Add the classroom member to the requestlist
  #if there is not a request from the same scource
  #still pending.

  set lnr [lsearch -glob $rg_req_iplist $ipnr]
  if {$lnr < 0} {

    #no request present: append to list.
    lappend rg_req_iplist $ipnr

    #store media too
    lappend rg_req_media_list $media

    #store userdata
    lappend rg_req_ud_list $userdata

    #request added: return true
    set result 1

  } else {

    #double request: just ignore for now.
    #but return a false indication.

   set result 0
  }

  return $result
}
proc rg:delete_from_reqlist {media ipnr} {

  global rg_req_iplist iplist rg_req_media_list rg_req_ud_list

  set m_left ""

  #find request
  set lnr [lsearch -glob $rg_req_iplist $ipnr]
  if {$lnr >= 0} {

    foreach m [rg:requested_media $ipnr] {
      if {[lsearch $media $m] < 0} {
        lappend m_left $m
      }
    }
```

```
    if {$m_left == ""} {
      #delete totally

      #delete from request lists
      set rg_req_iplist [lreplace $rg_req_iplist $lnr $lnr]
      #delete requested media from medialist
      set rg_req_media_list [lreplace $rg_req_media_list $lnr
$lnr]
      set rg_req_ud_list [lreplace $rg_req_ud_list $lnr $lnr]
    } else {
      #just delete requested media
      set rg_req_media_list [lreplace $rg_req_media_list $lnr
$lnr $m_left]
    }
  } else {

    #ignore error for now
  }
  return $m_left
}
proc rg:clear_reqlist {} {

  global rg_req_iplist rg_req_media_list rg_req_ud_list

  set rg_req_iplist ""
  set rg_req_media_list ""
  set rg_req_ud_list ""

}
proc rg:requested_media ipnr {

  global rg_req_iplist rg_req_media_list

  #find request
  set lnr [lsearch -glob $rg_req_iplist $ipnr]
  if {$lnr >= 0} {
    set media [lindex $rg_req_media_list $lnr]
  } else {
    set media ""
  }
  return $media
}
proc rg:add_to_grantlist {media ipnr} {

  global rg_grant_iplist rg_grant_media_list

  #keep track of media-grants

  lappend rg_grant_iplist $ipnr
  lappend rg_grant_media_list $media

}
proc rg:delete_from_grantlist {media ipnr} {

  global rg_grant_iplist rg_grant_media_list

  #keep track of media-grants

  set m_left ""

  #find grant number
  #delete from grant lists
  set lnr [lsearch -glob $rg_grant_iplist $ipnr]
  if {$lnr >= 0 } {

    foreach m [rg:granted_media $ipnr] {
      if {[lsearch $media $m] < 0} {
         lappend m_left $m
      }
    }
    if {$m_left == ""} {
      #delete totally
      set rg_grant_iplist [lreplace $rg_grant_iplist $lnr $lnr]
      set rg_grant_media_list [lreplace $rg_grant_media_list
$lnr $lnr]
    } else {
      #just delete requested media
      set rg_grant_media_list [lreplace $rg_grant_media_list
$lnr $lnr $m_left]
    }
  }

  return $m_left
}
proc rg:is_in_grantlist ipnr {

 global rg_grant_iplist

  #keep track of media-grants

    set lnr [lsearch -glob $rg_grant_iplist $ipnr]
    return "[expr $lnr >= 0]"

}
proc rg:clear_grantlist {} {

 global rg_grant_iplist rg_grant_media_list

  #clear media-grants

  set rg_grant_iplist ""
  set rg_grant_media_list ""

}
proc rg:granted_media ipnr {

  global rg_grant_iplist rg_grant_media_list

  #find request
  set lnr [lsearch -glob $rg_grant_iplist $ipnr]
  if {$lnr >= 0} {
    set media [lindex $rg_grant_media_list $lnr]
  } else {
    set media ""
  }
  return $media

}
proc rg:get_whole_grantlist {} {

  global rg_grant_iplist
  return $rg_grant_iplist
}
proc rg:get_requested_floors {} {

  global fp_requested

  return $fp_requested

}
proc rg:set_requested_floors f {

  global fp_requested

  set fp_requested $f
}
proc rg:set_granted_floors f {

  global fp_granted

  set fp_granted $f
}
proc rg:get_granted_floors {} {

  global fp_granted

  return $fp_granted

}
proc rg:add_requested_floors {} {

  global fp_requested

  lappend fp_requested $f

}
proc rg:add_granted_floors f {

  global fp_granted

  append fp_granted " $f"
}
proc rg:clear_granted_floors {} {

  global fp_granted

  set fp_granted ""

  return ""
}
proc rg:clear_requested_floors {} {

  global fp_requested

  set fp_requested ""

}
proc rg:req_or_all_media ipnr {

  set media  [rg:requested_media $ipnr]

  if {$media == ""} {
    #there was no reqeust: give all controlled media
    set media [fd:own_controlled_floors [fd:all_floor_ids]]
  }

  return $media
}
proc rg:get_ud_request ipnr {

  global rg_req_ud_list rg_req_iplist

  #find userdata
  set lnr [lsearch -glob $rg_req_iplist $ipnr]
  if {$lnr >= 0} {
    set media [lindex $rg_req_ud_list $lnr]
  } else {
    set media ""
  }
  return $media



}
#- TOP LEVEL procedures---------------------------------

set Name .floor_control
set Parent $Name
global Procs
set Procs($Name) { fc:return_floor fc:grant_floor
fc:grasp_floor fc:refuse_floor fc:grab_floor fc:floor_canceled
fc:floor_returned fc:floor_grabbed fc:floor_requested
fc:withdraw_f_request fc:floor_granted fc:floor_grasped
fc:floor_controlled fc:floor_refused fc:add_floor_user
fc:delete_floor_user fc:request_floor fc:floor_was_grabbed
fc:floor_given fc:floor_free fc:floors_not_controlled
fc:has_floorcontrol fc:set_controller}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) fc:
set Glob_prefix($Name) fc_
proc fc:return_floor media {

  #return access-tokens for floors in "media" to
  #floorcontroller.
```

```
  if {![fd:check_user_is_fc_for_floors 2 $media]} {
    fu:clear_tokens $media
    #return floor to owner
    main:status_message "Token returned for $media"
    set contr [fd:get_fcontrollers $media]
    foreach c $contr {
      nl:return_floor [fd:controlled_by $c $media] $c
    }
    return 1
  } else {
    return 0
  }
}
proc fc:grant_floor {media ipnr} {

  #grant access (give access-tokens) for floors in "media"
  #to "ipnr"

  if {[fd:check_user_is_fc_for_floors 1 $media]} {

    #give token
    nl:grant_floor $media $ipnr

    #status info
    main:status_message "Token granted for $media to $ipnr"

    return 1
  } else {
    return 0
  }
}
proc fc:grasp_floor media {

  #reset floors in "media" i.e.
  #grab access-tokens for floors in "media" from all users

  if {[fd:check_user_is_fc_for_floors 1 $media]} {

    #Grasp token: nobody except teacher may speak
    nl:grasp_floor $media

    #status info
    main:status_message "Resetting tokens for $media"

    return 1
  } else {
    return 0
  }
}
proc fc:refuse_floor {media ipnr userdata} {

  #refuse access-request for floors in "media"
  #to "ipnr".
  #An additional message can be given in "userdata".

  if {[fd:check_user_is_fc_for_floors 1 $media]} {

      main:status_message "Refusing token for $media to &ipnr"

      #make refusal known to the one requesting.
      nl:refuse_floor $media $ipnr $userdata
      return 1
  } else {
    return 0
  }
}
proc fc:grab_floor {media ipnr} {

  #grab access-tokens for floors in "media" from "ipnr"

 if {[fd:check_user_is_fc_for_floors 1 $media]} {

    main:status_message "Grabbing tokens for $media from $ipnr"

    #Grab token
    nl:grab_floor $media $ipnr

    return 1
  } else {
    return 0
  }
}
proc fc:floor_canceled {media ipnr} {

  #Accessrequest for floors in "media" by "ipnr"
  #was canceled.

  set media [fd:floors_exists $media]

 if {$media != ""} {
    main:status_message "Token-request for $media cancelled by
$ipnr"
    w_fc:floors_canceled $media $ipnr
 }
}
proc fc:floor_returned {media ipnr} {

  #Access-tokens for floors in "media" are returned
  #by "ipnr".

  set media [fd:floors_exists $media]

  if {$media != ""} {
    w_fc:floors_returned $media $ipnr
    main:status_message "Tokens returned for $media by $ipnr"
  }

}
proc fc:floor_grabbed media {

  #Access-tokens for floors in "media" are grabbed
  #by the floorcontroller.

  set media [fd:floors_exists $media]

  if {$media != ""} {
    #message
    main:status_message "Tokens grabbed for $media"

    set real [fu:clear_tokens $media]
    w_fc:floors_grabbed $real $media "Floor grabbed"
  }
}
proc fc:floor_requested {media ipnr userdata} {

  #Access requested for floors in "media" by "ipnr".
  #An additional message may be present in "userdata".

  set media [fd:floors_exists $media]
  if {$media != ""} {
    #message
    main:status_message "Token requested for $media by $ipnr"

    set my_media [fd:own_controlled_floors $media]
    w_fc:add_request $my_media $ipnr $userdata
  }
}
proc fc:withdraw_f_request media {

  #withdraw access-request to floorcontroller(s)
  #for floors in "media".

  if {![fd:check_user_is_fc_for_floors 2 $media]} {

    main:status_message "Withdrawing Tokenrequest for $media"

    #withdraw request for token
    set contr [fd:get_fcontrollers $media]
    foreach c $contr {
      nl:withdraw_request $media $c
    }
  }
}
proc fc:floor_granted {media ipnr} {

  #Access-token is granted for floors in "media" by "ipnr".
  #assumed is that "ipnr" is hereby the floorcontroller for
  #all floors in "media".

  set media [fd:floors_exists $media]

  if {$media != ""} {

    fc:set_controller $media $ipnr

    fu:set_tokens $media

    #message
    main:status_message "Floor granted for $media !"

    w_fc:floors_granted $media
  }
}
proc fc:floor_grasped {media ipnr} {

  #Access-token is grasped (grabbed from all participants)
  #for floors in "media" by "ipnr".
  #assumed is that "ipnr" is hereby the floorcontroller for
  #all floors in "media".

  set media [fd:floors_exists $media]

  if {$media != ""} {

   set real [fu:clear_tokens $media]

    #message
    main:status_message "Floor grasped for $media !"

    w_fc:floors_grabbed $real $media "Floor grasped"
   fc:set_controller $media $ipnr

 }
}
proc fc:floor_controlled {media ipnr} {

  #Register all floors & directions in "media" as
  #controlled. The controller is at "ipnr".

  set media [fd:floors_exists $media]

  if {$media != ""} {
    #if floors are alredy controlled by this user
    #then keep them from being grabbed
    set m [fd:user_is_not_controller_for $media]
    if {$m != $media} {
      main:status_message "ERROR: Controller conflict for
[fd:own_controlled_floors $media] !!!"
    }

   main:status_message "$ipnr [ud:name_in_meetinglist $ipnr 1]
is controller now for $media"

   set first [fd:set_fcontroller $m $ipnr]
   fu:redef_floors $m 0 0

   w_fc:controller_status_change $media

   w_fc:snew_floorcontroller $first [fd:id $m] $ipnr
  }
}
proc fc:floor_refused {media ipnr userdata} {
```

```
  #Access-request is refused for floors in "media" by
  #the controller at "ipnr".
  #An additional message may be present in "userdata".

  set media [fd:floors_exists $media]
  if {$media != ""} {

    main:status_message "Tokenrequest was refused for $media"

    w_fc:floors_refused $media $ipnr $userdata
  }
}
proc fc:add_floor_user {ipnr uid} {

  #Register a new floor-user.

   set nr [ud:get_usernumber $uid]
   if {$nr >= 0} {
     set name [ud:get_username $nr 0]
   } else {
     set name $uid
   }

   #message
   main:status_message "$ipnr joins meeting"

   #pass this to the HCI
   w_fc:add_participant $ipnr $name
}
proc fc:delete_floor_user ipnr {

  #unregister a floor-user

  #message
  main:status_message "$ipnr leaves meeting"

  #pass to HCI
  w_fc:delete_participant $ipnr

}
proc fc:request_floor {media userdata} {

  #Request access to floorcontroller(s)
  #for floors in "media".

  if {![fd:check_user_is_fc_for_floors 2 $media]} {
    if {[fd:has_controller 1 $media]} {

      #Request floor

      #request token from controllers
      set contr [fd:get_fcontrollers $media]
      foreach c $contr {
        nl:request_floor $media $c $userdata
      }
      #status info
      set er_message "Token requested for $media"
      return 1
    } else {
      return 0
    }
  } else {
    return 0
  }
}
proc fc:floor_was_grabbed {media ipnr} {

  #Confirmation: Access-token was succesfully grabbed
  #for floors in "media" from "ipnr".

  set media [fd:floors_exists $media]
  if {$media != ""} {

    if {[w_fc:grab_floors_done $media $ipnr]} {
      #Floor grab successful
      #status info
      main:status_message "Token was grabbed from $ipnr for
$media"
    }
  }
}
proc fc:floor_given {media ipnr} {

  set media [fd:floors_exists $media]

  if {$media != ""} {

    #status info
    main:status_message "Token given for $media"

    fc:floor_returned [fd:own_controlled_floors $media] $ipnr
    fc:floor_granted [fd:not_own_controlled_floors $media]
$ipnr
  }
}
proc fc:floor_free media {

  set media [fd:floors_exists $media]

  if {$media != ""} {

    #status info
    main:status_message "Flooraccess is free for $media"

    fd:set_fcontroller $media U
    fu:redef_floors $media 0 0
    w_fc:controller_status_change $media
  }
}
proc fc:floors_not_controlled media {

  set media [fd:floors_exists $media]

  if {$media != "" } {
    #status info
    main:status_message "Floor is uncontrolled for $media"

    fd:set_fcontroller $media ""
    set real [fu:clear_tokens $media]
    w_fc:floors_grabbed $real $media "Floor uncontrolled"
    fu:redef_floors $media 0 0
  }
  w_fc:controller_status_change $media
}
proc fc:has_floorcontrol {grasp media} {

  set media [fd:floors_exists $media]
  if {$media != ""} {

    #status info
    main:status_message "You are floorcontroller for $media"

    #register control
    fd:set_fcontroller $media X
    fu:redef_floors $media 0 0

    #let others know (if this function was not enabled, this
    #procedure will do nothing)
    set c_floors [fd:own_controlled_floors [fd:all_floor_ids]]

    w_fc:controller_status_change $media

    #start_announce_control $media

    if {$grasp} {
      fc:grasp_floor $media
    }
  }
}
proc fc:set_controller {media ipnr} {

  if {$media != "" && [fd:get_fcontrollers $media] != $ipnr} {
    #controller properties have changed
    #so include those.
    #this is to be used in the testmode
    #because this is the only way for a controller to be
    #known.

    fc:floor_controlled $media $ipnr
  }
}
#- TOP LEVEL procedures---------------------------------

set Name .main
set Parent $Name
global Procs
set Procs($Name) {main main:give_usage main:status_message
main:save_winpos}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) main:
set Glob_prefix($Name) main_
proc main {} {

  #import extra modules
  #uplevel #0 {source ms_ihci.tcl}

  set ex ""
    #Executable for sessionlayer
  set u_nr ""
    #user number
  set m_nr ""
    #meeting number
  set no_c 0
    #no camera persent
  set u_atm 0
    #Use IP over ATM
  set dbg 0
    #Use Debug-window
  set vcoding "CellB"
    #Video Codec
  set vsize 1
    #Video Size
  set vpreview 1
    #Video preview
  set pict 1
    #Show Pictures (Faces-extension)
  set no_mm 0
  set sl_emu 0

  #read commandline parameters
  set pars [sy:get_cmdln_params]

  set lnr [lsearch -glob $pars "-H"]
  if {$lnr >= 0} {
    #give useage this DTC system
    main:give_usage
  }

  set lnr [lsearch -glob $pars "-h"]
  if {$lnr >= 0} {
    #give useage this DTC system
    main:give_usage
  }

set lnr [lsearch -glob $pars "-L"]
  if {$lnr >= 0} {
  set no_mm 1
  }

  set lnr [lsearch -glob $pars "-E"]
  if {$lnr >= 0} {
    #user_number given ?
    set ex [lindex $pars [expr $lnr + 1]]
  }
```

```
  sy:define $ex
  sy:def_images

  set lnr [lsearch -glob $pars "-U"]
  if {$lnr >= 0} {
    #user_number given ?
    set u_nr [lindex $pars [expr $lnr + 1]]
  }

  set lnr [lsearch -glob $pars "-M"]
  if {$lnr >= 0} {
    #meeting-number given ?
    set m_nr [lindex $pars [expr $lnr + 1]]
    if {[catch {set dummy [expr $m_nr + 1]}]} {
      puts "No number for -M"
      main:give_usage
    }
  }

  set lnr [lsearch -glob $pars "-N"]
  if {$lnr >= 0} {
    #user has no camera
    set no_c 1
  }

  set lnr [lsearch -glob $pars "-A"]
  if {$lnr >= 0} {
    #use atm
    set u_atm 1
  }

  set lnr [lsearch -glob $pars "-D"]
  if {$lnr >= 0} {
    #debug on
    set dbg 1
  }

  set lnr [lsearch -glob $pars "-F"]
  if {$lnr >= 0} {
    #faces off
    set pict 0
  }

  set lnr [lsearch -glob $pars "-C"]
  if {$lnr >= 0} {
    #video coding given ?
    set vcoding [lindex $pars [expr $lnr + 1]]
    if {$vcoding != "CellB" && $vcoding != "Jpeg" && $vcoding
!= "UYVY" && $vcoding != "Mpeg1"} {
      puts "-C has wrong parameter"
      main:give_usage
    }
  }

  set lnr [lsearch -glob $pars "-S"]
  if {$lnr >= 0} {
    #video-size given ?
    set vsize [lindex $pars [expr $lnr + 1]]
    if {[catch {set dummy [expr $vsize + 1]}]} {
      puts "no number for -S"
      main:give_usage
    } else {
      if {($vsize < 0) || ($vsize > 4)} {
        puts "-S out of range"
        main:give_usage
      }
    }
  }

  set lnr [lsearch -glob $pars "-P"]
  if {$lnr >= 0} {
    #debug on
    set vpreview 0
  }

  set lnr [lsearch -glob $pars "-X"]
  if {$lnr >= 0} {
    #session layer emulation
    set sl_emu 1
    #no local media whatsoever?
    #set no_c 1
    #set no_mm 1
    #set vpreview 0
  }

  set lnr [lsearch -glob $pars "-T"]
  if {$lnr >= 0} {
    #testmode
    w_st1:init $m_nr $u_nr $no_c $u_atm $dbg $vcoding $vsize
$vpreview $pict $sl_emu
  } else {
    set lnr [lsearch -glob $pars "-Q"]
    if {$lnr >= 0} {
      #use database, start now
      if {$u_nr != "" && $m_nr != ""} {
        w_st2:init 1 $m_nr $u_nr $no_c $u_atm $dbg $vcoding
$vsize $vpreview $pict #sl_emu
      } else {
        main:give_usage
      }
    } else {
      w_st2:init 0 $m_nr $u_nr $no_c $u_atm $dbg $vcoding
$vsize $vpreview $pict $sl_emu
    }
  }
}
proc main:give_usage {} {

  puts ""
  puts ""
  puts "Usage of dtc system:"
  puts ""
  puts "-H        : This message"
  puts ""
  puts "-L        :no local media"
  puts "-M nr     : Meeting number"
  puts "-U n     : User name or number"
  puts "-Q        : Quickstart without login. This requires
that a meeting nr and user-nr are give"
  puts "            This will be ignored with the -T option."
  puts ""
  puts "-C coding : Video-coding. Must be CellB, Jpeg, UYVY or
Mpeg1. CellB is default."
  puts "-N        : No camara available"
  puts "-S        : Size for video-out. Must be between 0 and
4. Default = 2"
  puts "-P        : Preview Off."
  puts "-F        : Faces disabled"
  puts ""
  puts "-A        : Use ATM"
  puts ""
  puts "-D        : Debugging-mode on (is default on in
testmode)"
  puts "-T        : Testmode: This means that another
startwindow is used"
  puts "-E        : Executable for networklayer"
  puts "-X        : Emulation of networklayer: this means also
no local media"
  puts ""

  sy:quit_all
}
proc main:status_message s {

  global er_message

  catch {
    w_sc:addlst_debug "Status: $s"
  }

  set er_message $s
  update idletasks

}
proc main:save_winpos {} {

  #save all windows-positions in the position-file
  #This use procedures that are standard present in
  #common.tcl or common2.tcl

  global posfilename

  #gather all windows-info
  defaultpos_prepsave_all

  #save it
  defaultpos_save $posfilename

}
#- TOP LEVEL procedures---------------------------------

set Name .floor_use
set Parent $Name
global Procs
set Procs($Name) { fu:clear_token fu:set_token fu:set_mute
fu:clear_mute fu:enable_floor_write fu:enable_floor_read
fu:disable_floor_write fu:disable_floor_read fu:text_out
fu:text_in fu:delete_floor fu:set_tokens fu:clear_tokens
fu:set_audiofloor_volume fu:redef_floors fu:add_video_read
fu:delete_video_read fu:video_write_unusable
fu:video_read_unusable fu:audio_unusable}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) fu:
set Glob_prefix($Name) fu_
proc fu:clear_token {dir f} {

   if {[fd:check_user_is_fc_for_floors 0 $f]} {
     #no clear_token when user is controller
     return
   }

   main:status_message "clearing token for $dir$f"

   if {$dir != "R"} {
     #write not permitted
     if {[fd:has_token W $f]} {
       fd:set_token W $f 0
       fu:disable_floor_write $f
     }
   }
   if {$dir != "W"} {
     #read not permitted
     if {[fd:has_token R $f]} {
       fd:set_token R $f 0
       fu:disable_floor_read $f
     }
   }
}
proc fu:set_token {dir f} {

   main:status_message "setting token for $dir$f"

   if {$dir != "R"} {
     #write permitted
     if {![fd:has_token W $f]} {
       fd:set_token W $f 1
       fu:enable_floor_write $f 0
     }
   }
   if {$dir != "W"} {
     #read permitted
     if {![fd:has_token R $f]} {
       fd:set_token R $f 1
```

```
        fu:enable_floor_read $f 0
      }
    }
}
proc fu:set_mute {dir f} {

   main:status_message "muting $dir$f"

   if {$dir != "R"} {
     #write not permitted
     if {[fd:is_muted W $f] == 0} {
       fd:set_mute W $f 1
       fu:disable_floor_write $f
     }
   }
   if {$dir != "W"} {
     #read not permitted
     if {[fd:is_muted R $f] == 0} {
       fd:set_mute R $f 1
       fu:disable_floor_read $f
     }
   }
}
proc fu:clear_mute {dir f} {

  set result 0

   main:status_message "un-muting $dir$f"

   if {$dir != "R"} {
     #write permitted
     if {[fd:is_muted W $f] == 1} {
       fd:set_mute W $f 0
       fu:enable_floor_write $f 0
       set result 1
     }
   }
   if {$dir != "W"} {
     #read permitted
     if {[fd:is_muted R $f] == 1} {
       fd:set_mute R $f 0
       fu:enable_floor_read $f 0
       set result 1
     }
   }

   return $result
}
proc fu:enable_floor_write {f ind} {

  if {[fd:has_token W $f]} {
    if {![fd:is_muted W $f]} {
      main:status_message "enabling write for $f"
      switch -glob -- [fd:get_floortype $f] {
        A {
            nl:enable_au_write
        }
        V {
            nl:enable_vid_write
        }
        T {
          #do nothing: text only needs status chenge
        }
      }
      return 1
    } else {
      return 0
    }
  } else {
    if {$ind} {
     main:status_message "ERROR: you have no right"
    }
    return 0
  }

}
proc fu:enable_floor_read {f ind} {

  if {[fd:has_token R $f]} {
    if {![fd:is_muted R $f]} {
      main:status_message "enabling read for $f"
      switch -glob -- [fd:get_floortype $f] {
        A {
            nl:enable_au_read
        }
        V {
            nl:enable_all_video_read
        }
        T {
          #do nothing: text only needs status chenge
        }
      }
      return 1
    } else {
      return 0
    }
  } else {
    if {$ind} {
     main:status_message "ERROR: you have no right"
    }
    return 0
  }

}
proc fu:disable_floor_write f {

  main:status_message "disabling write for $f"
  switch -glob -- [fd:get_floortype $f] {
    A {
        nl:disable_au_write
    }
    V {
        nl:disable_vid_write
    }
    T {
      #do nothing: text only needs status chenge
    }
  }
}
proc fu:disable_floor_read f {

  main:status_message "disabling read for $f"
  switch -glob -- [fd:get_floortype $f] {
    A {
        nl:disable_au_read
    }
    V {
        nl:disable_all_video_read
    }
    T {
      #do nothing: text only needs status chenge
    }
  }
}
proc fu:text_out {channel text} {

  set f [fd:id_from_channeltype $channel T]
  if {[fd:has_token W $f]} {
    if {![fd:is_muted W $f]} {
      nl:broadcast_text $channel $text
      main:status_message "Sending text"
      return 1
    } else {
      return 0
    }
  } else {
   main:status_message "ERROR: you have no right"
    return 0
  }
}
proc fu:text_in {channel ipnr text} {

  set f [fd:id_from_channeltype $channel T]
  if {$f == ""} {
     return
  }

  if {[fd:has_token R $f]} {
    if {![fd:is_muted R $f]} {
       w_txt:display_incoming $channel $ipnr $text
    }
  }
}
proc fu:delete_floor f {

  fu:clear_token X $f

}
proc fu:set_tokens media {

  foreach m $media {

    if {$m != ""} {
       set dir [fd:dir $m]
       fu:set_token $dir [fd:id $m]
    }
  }
}
proc fu:clear_tokens media {

  set real 0

  foreach m $media {

    if {$m != ""} {
       set dir [fd:dir $m]
       set f [fd:id $m]
       set real [expr {$real || [fd:has_token $dir $f]}]
       fu:clear_token $dir $f
    }
  }
  return $real
}
proc fu:set_audiofloor_volume {channel source_ips value} {

  #set volume for an audio floor given by "channel"
  #to "value" for all sources as given by "source_ips".

  #For now, since there is only one channel,
  #"ch" is ignored.

  foreach ip $source_ips {
    nl:set_mixing_volume $ip $value
  }
}
proc fu:redef_floors {m fl_w_token_t fl_r_token_t} {

 foreach f [fd:id $m] {
   if {[fd:check_user_is_fc_for_floors 0 $f] || [fd:is_free
$f]} {
      #is controller or uncontrolled ?
      #has token for his own floor
      fu:set_token X $f
    } else {
      #take first definitions
      if {$fl_w_token_t || [fd:controldir $f] == "R"} {
        fu:set_token W $f
      } else {
        fu:clear_token W $f
      }
      if {$fl_r_token_t || [fd:controldir $f] == "W"} {
        fu:set_token R $f
      } else {
        fu:clear_token R $f
```

```
      }
    }
  }
}
proc fu:add_video_read {channel source_ips} {

  foreach ip $source_ips {
     nl:add_video_source $ip
  }
}
proc fu:delete_video_read {channel source_ips} {

  set result ""

  foreach ip $source_ips {
     lappend result [nl:delete_video_source $ip 1]
  }

  return $result
}
proc fu:video_write_unusable {} {

 #video write is inpossible, so make floors unusable
 foreach chn [fd:all_channels_for_floortype V] {
   set m [fd:id_from_channeltype $chn V]
   fd:set_unusable W $m
   fh:set_unusable W $m
 }

}
proc fu:video_read_unusable {} {

 #video read is inpossible, so make floors unusable
 foreach chn [fd:all_channels_for_floortype V] {
   set m [fd:id_from_channeltype $chn V]
   fd:set_unusable R $m
   fh:set_unusable R $m
 }
}
proc fu:audio_unusable {} {

 #video write is inpossible, so make floors unusable
 foreach chn [fd:all_channels_for_floortype A] {
   set m [fd:id_from_channeltype $chn A]
   fd:set_unusable W $m
   fd:set_unusable R $m
   fh:set_unusable X $m
 }
}
#- TOP LEVEL procedures---------------------------------

set Name .floor_def
set Parent $Name
global Procs
set Procs($Name) { fd:set_fcontroller
fd:check_user_is_fc_for_floors fd:get_fcontrollers
fd:own_controlled_floors fd:not_own_controlled_floors
fd:controlled_by fd:all_floor_ids fd:name
fd:get_floors_controlled_by fd:get_floortype fd:id
fd:controldir fd:dir fd:exists fd:add_floor fd:channel
fd:delete_floor fd:all_channels_for_floortype
fd:id_from_channeltype fd:has_controller fd:reset_floors
fd:is_free fd:set_token fd:has_token fd:set_mute fd:is_muted
fd:floors_exists fd:user_is_not_controller_for fd:has_tokens
fd:set_unusable}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) fd:
set Glob_prefix($Name) fd_
proc fd:set_fcontroller {media c_ipnr} {

  #Register a new floorcontroller for all floors in "media".
  #return 1 (true) if this was the first known controller

  global fd_floorids fd_control fd_control_dir

  set mf [fd:id $media]

  #Was there a controller ?
  set i 1
  foreach f $fd_floorids {
    if {$fd_control($f) != "U" && $fd_control($f) != "" &&
$fd_control($f) != "X"} {
      set i 0
      break
    }
  }

  #store controller address/type
  foreach f $media {
    set fl [fd:id $f]
    set fd_control($fl) $c_ipnr
    set fd_control_dir($fl) [fd:dir $f]
  }
  return $i
}
proc fd:check_user_is_fc_for_floors {m media} {

  global fd_control

  #check if user is floor controller for all floors in media
  #M=1: give message when not controller
  #M=2: give message when controller

  set mf [fd:id $media]

  if {$mf == ""} {
    set i 0
  } else {

    set i 1
    foreach f $mf {
      if {$fd_control($f) != "X"} {
        set i 0
        break
      }
    }
  }

  if {!$i && $m == 1} {
    main:status_message "ERROR: you have no rights for that
!!!"
  }
  if {$i && $m == 2} {
    main:status_message "ERROR: you are floor controller !!!"
  }

  return $i

}
proc fd:get_fcontrollers media {

  global  fd_control

  set mf [fd:id $media]

  set contr ""
  foreach f $mf {
    set c $fd_control($f)
    if {$c != "U" && $c != "X" && $c != "" && [lsearch $contr
$c] == -1} {
      lappend contr $c
    }
  }
  return $contr
}
proc fd:own_controlled_floors media {

  global  fd_floorids fd_control fd_type fd_name fd_token
  global  fd_own_floors fd_free_floors fd_uncontr_floors

  set new_m ""

  set mf [fd:id $media]

  foreach f $mf {
    if {$fd_control($f) == "X"} {
      lappend new_m "[fd:controldir $f]$f"
    }
  }
  return $new_m
}
proc fd:not_own_controlled_floors media {

  global  fd_floorids fd_control fd_type fd_name fd_token

  set new_m ""

 set mf [fd:id $media]

  foreach f $mf {
    if {$fd_control($f) != "X" && $fd_control($f) != "" &&
$fd_control($f) != "U"} {
      lappend new_m "[fd:controldir $f]$f"
    }
  }
  return $new_m
}
proc fd:controlled_by {ipnr media} {

  global fd_floorids fd_control

  set m ""
  foreach f [fd:id $media] {
    if {$fd_control($f) == "$ipnr"} {
      lappend m $f
    }
  }
  return $m
}
proc fd:all_floor_ids {} {

  global fd_floorids

  return $fd_floorids
}
proc fd:name f {

  global fd_name

return $fd_name($f)

}
proc fd:get_floors_controlled_by c {

  global fd_floorids fd_control

  set c_floors ""

  foreach f $fd_floorids {
    if {$c == $fd_control($f)} {
      lappend c_floors "[fd:controldir $f]$f"
    }
  }
  return $c_floors
}
proc fd:get_floortype f {

  global fd_type

  return $fd_type($f)

}
proc fd:id media {
```

```
  set new_media ""

  foreach f $media {
    if {$f != ""} {
      if {[string match {[RWX]} [string index $f 0]]} {
        set f [string range $f 1 end]
      }
      lappend new_media $f
    }
  }
  return $new_media
}
proc fd:controldir f {

  global fd_control_dir

  return $fd_control_dir($f)
}
proc fd:dir m {

  set dir [string index $m 0]
  if {![string match {[RWX]} $dir] } {
    #no direction indicated: assume write
    set dir "W"
  }
  return $dir
}
proc fd:exists f {

  global fd_floorids

  set i 0

  foreach fl $fd_floorids {
    if {$fl == $f} {
      set i 1
      break
    }
  }
  return $i
}
proc fd:add_floor {f fl_type_t fl_channel_t fl_name_t
fl_w_mute_t fl_r_mute_t} {

  global  fd_floorids fd_control fd_control_dir fd_type
fd_channel fd_name
  global fd_w_mute fd_r_mute fd_w_token fd_r_token

  lappend fd_floorids $f

  set fd_control($f) ""
  set fd_control_dir($f) "X"
  set fd_type($f) $fl_type_t
  set fd_channel($f) $fl_channel_t
  set fd_name($f) $fl_name_t
  set fd_w_token($f) 0
  set fd_r_token($f) 0
  set fd_w_mute($f) $fl_w_mute_t
  set fd_r_mute($f) $fl_r_mute_t
}
proc fd:channel f {

  global fd_channel

  return $fd_channel($f)

}
proc fd:delete_floor f {

  global  fd_control fd_control_dir fd_type fd_channel fd_name
fd_floorids
  global  fd_w_token fd_r_token fd_w_mute fd_r_mute

  #no more reception/transmission

  fu:delete_floor $f

  unset fd_control($f)
  unset fd_control_dir($f)
  unset fd_type($f)
  unset fd_channel($f)
  unset fd_name($f)
  unset fd_w_token($f)
  unset fd_r_token($f)
  unset fd_w_mute($f)
  unset fd_r_mute($f)


  set fd_floorids [array names fd_control]
}
proc fd:all_channels_for_floortype type {

  global  fd_floorids fd_control fd_type fd_channel

  set ch_all ""

  foreach f $fd_floorids {
    if {$fd_type($f) == $type} {
      lappend ch_all $fd_channel($f)
    }
  }

  return $ch_all
}
proc fd:id_from_channeltype {channel type} {

  global  fd_floorids fd_type fd_channel fd_name

  set found 0

  foreach f $fd_floorids {
    if {$fd_type($f) == $type} {
      if {$channel == $fd_channel($f)} {
        set found 1
        break
      }
    }
  }
  if {$found} {
    return $f
  } else {
    return ""
  }
}
proc fd:has_controller {m media} {

 global fd_control

  #check if floors have no controller for all floors in media
  #M=1: give message

  set mf [fd:id $media]

  set i 0
  foreach f $mf {
    if {($fd_control($f) != "") && ($fd_control($f) != "U")  &&
($fd_control($f) != "X")} {
      set i 1
      break
    }
  }

  if {!$i && $m} {
    main:status_message "ERROR: floor controller unknown !!!"
  }
  return $i

}
proc fd:reset_floors {} {

  global  fd_floorids fd_control fd_control_dir fd_type
fd_channel fd_name
  global  fd_w_token fd_r_token fd_w_mute fd_r_mute
  global floor_to_ch

  set fd_floorids ""

  catch "unset floor_to_ch"

  catch "unset fd_control"
  catch "unset fd_control_dir"
  catch "unset fd_type"
  catch "unset fd_channel"
  catch "unset fd_name"
  catch "unset fd_w_token"
  catch "unset fd_r_token"
  catch "unset fd_w_mute"
  catch "unset fd_r_mute"

}
proc fd:is_free f {

  global fd_control

  #check if floor is uncontrolled (free)

  return [expr {$fd_control($f) == "U"}]

}
proc fd:set_token {dir f value} {

  global fd_w_token fd_r_token

   if {$dir == "W"} {
      set fd_w_token($f) $value
   }

   if {$dir == "R"} {
      set fd_r_token($f) $value
   }
}
proc fd:has_token {dir f} {

  global fd_w_token fd_r_token

  if {$dir == "W"} {
    return $fd_w_token($f)
  }
  if {$dir == "R"} {
    return $fd_r_token($f)
  }
   if {$dir == "X"} {
    set t [expr {$fd_r_token($f) ==1 && $fd_w_token($f) ==1}]
    return $t
  }
}
proc fd:set_mute {dir f value} {

  global fd_w_mute fd_r_mute

   if {$dir == "W"} {
      set fd_w_mute($f) $value
   }
   if {$dir == "R"} {
      set fd_r_mute($f) $value
   }
}
proc fd:is_muted {dir f} {

  global fd_w_mute fd_r_mute

   if {$dir == "R"} {
     return $fd_r_mute($f)
   }
```

```
   if {$dir == "W"} {
     return $fd_w_mute($f)
   }

}
proc fd:floors_exists media {

  global fd_floorids

  set result ""

  foreach m $media {
    if {[fd:exists [fd:id $m]]} {
      lappend result $m
    }
  }
  return $result
}
proc fd:user_is_not_controller_for media {

  global fd_floorids fd_control

  set new_m ""

  foreach m $media {
    if {$fd_control([fd:id $m]) != "X"} {
      lappend new_m "$m"
    }
  }
  return $new_m
}
proc fd:has_tokens media {

  set result 0

  foreach m $media {
     if {[fd:has_token [fd:dir $m] [fd:id $m]]} {
       set result 1
       break
     }
   }

  return $result
}
proc fd:set_unusable {dir f} {

   #no disable needed because floor was unusable

   if {$dir != "R"} {
     #write not permitted
     fd:set_mute W $f -1
   }
   if {$dir != "W"} {
     #read not permitted
     fd:set_mute R $f -1
   }

}
#- TOP LEVEL procedures---------------------------------

set Name .floor_read_db
set Parent $Name
global Procs
set Procs($Name) {fr:read_meetingdbase fr:find_rowfortype
fr:current_mname fr:read_template_dbase fr:get_reflector
fr:get_all_meetings_info fr:get_one_meeting_name
fr:set_floor_db fr:check_for_controller fr:get_actor_indication
fr:current_mnr}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) fr:
set Glob_prefix($Name) fr_
proc fr:read_meetingdbase {mynr meetingnr} {

  global fr_actor_indication fr_conference_name fr_mnr
  global fr_read_db
  global fr_actor_type fr_actor_name fr_floor fr_mnr
  global fr_fc_role fr_fp_role fr_gp_role fr_gc_role

  #read groupmeeting definition databases
  #find meetingnumber and userid and
  #define all floors accordingly

  set fr_mnr $meetingnr

  set fr_read_db 1

  fr:get_one_meeting_name $meetingnr

  set fp_m [sy:open_db_file "tele_m.ini"]

  set mrow [fr:find_rowfortype $fp_m M $meetingnr]

  set mtype [lindex $mrow 2]
  fr:read_template_dbase $mtype

  set mrow [sy:read_db_line $fp_m]
  while {[lindex $mrow 0] == "P"} {
    set fr_actor_type([lindex $mrow 1]) [lindex $mrow 2]
    set mrow [sy:read_db_line $fp_m]
  }

  sy:close_db_file $fp_m

  fd:reset_floors

  #set all initial floordefinitions
  set t 0

  if {[catch {set my_role $fr_actor_type($mynr)}]} {
    main:status_message "ERROR: User has no role !"
    return 0
  }

  if {($fr_gc_role == 0) && ($fr_gp_role($my_role) == 0)} {
    main:status_message "ERROR: User may not enter groupmeeting
!"
    return 0
  }

  main:status_message "Role: $fr_actor_name($my_role)"

  foreach f [array names fr_floor] {
   #is floor controller or participant ?
   if {([lsearch -exact [fd:id $fr_fp_role($my_role)] $f] >= 0)
|| ([lsearch -exact [fd:id $fr_fc_role($my_role)] $f] >= 0)} {
    set fl_type_t [string range $f 0 0]
    set fl_channel_t [string range $f 1 1]
    set fl_name_t $fr_floor($f)
    set fl_r_mute_t 0
    set fl_w_mute_t 0
    fh:add_floor $f $fl_type_t $fl_channel_t $fl_name_t
$fl_r_mute_t $fl_w_mute_t

    #is floor free ?
    set ctrl 0
    foreach a [array names fr_actor_name] {
      set i [lsearch -exact [fd:id $fr_fc_role($a)] $f]
      if {$i >= 0} {
        set ctrl 1
        break
      }
    }

    if {!$ctrl} {
      fc:floor_free $f
    } else {
      #check for floor-control
      set my_floors $fr_fc_role($my_role)
      set i [lsearch -glob [fd:id $my_floors] $f]
      if {$i >= 0} {
        set cdir [fd:dir [lindex $my_floors $i]]
        fc:has_floorcontrol 0 "$cdir$f"
      } else {
        #no floor tokens yet. These will be set when the
        #controller is added to the meeting
        fc:floors_not_controlled $f
      }
    }
   }
  }
  set fr_actor_indication $fr_actor_name($my_role)

  return 1
}
proc fr:find_rowfortype {f rtype id} {

  set ok 1
  while {$ok} {
    set ln [sy:read_db_line $f]
    if {$ln == "END."} {
      set ok 0
      set result ""
    } elseif {[lindex $ln 0] == $rtype} {
      if {[lindex $ln 1] == $id} {
        set ok 0
        set result $ln
      }
    }
  }
  return $result
}
proc fr:current_mname {} {

  #return conference name

  global fr_conference_name

  return $fr_conference_name
}
proc fr:read_template_dbase mtype {

  global fr_actor_name fr_floor fr_fc_role fr_fp_role
fr_gc_role fr_gp_role

  catch {unset fr_actor_name}
  catch {unset fr_floor}
  catch {unset fr_fc_role}
  catch {unset fr_fp_role}
  catch {unset fr_gp_role}
  set fr_gc_role ""

  set f [sy:open_db_file "tele_mt.ini"]

  #find type (this has no error-checks)
  set row [fr:find_rowfortype $f T $mtype]

  #find A (actor) rows
  set row [sy:read_db_line $f]
  while {[lindex $row 0] == "A"} {
    set a [lindex $row 1]
    set fr_actor_name($a) [lindex $row 2]
    set fr_fc_role($a) ""
    set fr_fp_role($a) ""
    set fr_gp_role($a) 0
    set row [sy:read_db_line $f]
  }

  #find F (fr_floor) rows
  while {[lindex $row 0] == "F"} {
    set fr_floor([lindex $row 1]) [lindex $row 2]
    set row [sy:read_db_line $f]
  }

  #find R (role) rows
```

```
  while {[lindex $row 0] == "R"} {
    if {[lindex $row 2] == "FC"} {
      append fr_fc_role([lindex $row 1]) [lrange $row 3 end]
    } else {
      if {[lindex $row 2] == "FP"} {
        append fr_fp_role([lindex $row 1]) [lrange $row 3 end]
      } else {
        if {[lindex $row 2] == "GC"} {
          set fr_gc_role [lindex $row 1]
        } else {
          if {[lindex $row 2] == "GP"} {
            set fr_gp_role([lindex $row 1]) 1
          }
        }
      }
    }
    set row [sy:read_db_line $f]
  }


  sy:close_db_file $f
}
proc fr:get_reflector meetingnr {

  set fp_m [sy:open_db_file "tele_m.ini"]

  set refl_adr ""

  set mrow [fr:find_rowfortype $fp_m M $meetingnr]

  if {$mrow != ""} {
    set refl_adr [lindex $mrow 3]
  }

  sy:close_db_file $fp_m

  main:status_message "Reflector: $refl_adr"

  return $refl_adr
}
proc fr:get_all_meetings_info {} {

  #return list with alternating meetingnumbers and -names

  set f [sy:open_db_file "tele_mn.ini"]

  set mnumber ""

  set ok 1
  while {$ok} {
    set ln [sy:read_db_line $f]
    if {$ln == "END."} {
      set ok 0
    } else {
      lappend mnumber [lindex $ln 0] [lindex $ln 1]
    }
  }

  sy:close_db_file $f

  return $mnumber

}
proc fr:get_one_meeting_name meetingnr {

  global fr_conference_name

  set f [sy:open_db_file "tele_mn.ini"]

  set row [sy:read_db_line $f]
  while {$row != "END."} {
    if {[lindex $row 0] == $meetingnr} {
      set fr_conference_name [lindex $row 1]
      break
    }
    set row [sy:read_db_line $f]
  }

  sy:close_db_file $f

  main:status_message "Meeting: $fr_conference_name"

  return $fr_conference_name
}
proc fr:set_floor_db {no_mm vid_uncontrol is_teacher r_adr} {

  global fr_actor_indication
  global fr_conference_name fr_nmr fr_mnr
  global fr_read_db

  set fr_read_db 0

  #should come from a DB

  if {$no_mm} {
    set floors "T1 T2"
    set floornames "{Text Channel} {Chat}"
  } else {
    set floors "A1 V1 T1"
    set floornames "{Audio} {Video} {Text Channel}"
  }

  set fr_conference_name $r_adr
  set fr_mnr 0


  fd:reset_floors

  #set all initial floordefinitions
  set t 0
  foreach f $floors {
    set fl_type_t [string range [lindex $floors $t] 0 0]
    set fl_channel_t [string range [lindex $floors $t] 1 1]
    set fl_name_t [lindex $floornames $t]
    #start muted
    set fl_r_mute_t 1
    set fl_w_mute_t 1

    set fl_control_dir_t "W"

    fh:add_floor $f $fl_type_t $fl_channel_t $fl_name_t
$fl_r_mute_t $fl_w_mute_t

    #is video free ?
    if {$vid_uncontrol && ($fl_type_t == "V")} {
      fc:floors_free "X$f"
    } else {
      #for now: control all floors when you are teacher
      if {$is_teacher} {
        fc:has_floorcontrol 1 "W$f"
      } else {
       fc:floors_not_controlled $f
      }
    }

    incr t
  }

  if {$is_teacher} {
    set fr_actor_indication "TEACHER"
  } else {
    set fr_actor_indication "Student"
  }
}
proc fr:check_for_controller id {

  global fr_read_db
  global fr_actor_type fr_fc_role fr_fp_role

  if {$fr_read_db} {

    set nr [ud:get_usernumber $id]

    #is role defined ?
    if {[catch {set act $fr_actor_type($nr)}]} {
      set media ""
    } else {
      set media $fr_fc_role($act)
    }
    return $media

  } else {

     return ""
  }
}
proc fr:get_actor_indication {} {

  global fr_actor_indication

  return $fr_actor_indication

}
proc fr:current_mnr {} {

  #return conference number

  global fr_mnr

  return $fr_mnr
}
#- TOP LEVEL procedures---------------------------------

set Name .floor_hci
set Parent $Name
global Procs
set Procs($Name) {fh:create_floor_win fh:destroy_floor_win
fh:set_message fh:set_init_message fh:add_floor fh:delete_floor
fh:delete_all_floors fh:set_messages fh:audio_unusable
fh:video_write_unusable fh:get_full_floormanes
fh:video_read_unusable fh:set_unusable}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) fh:
set Glob_prefix($Name) fh_
proc fh:create_floor_win {type channel name} {

  switch -glob -- $type {

    A {
      #only one floor possible
      #audio: no window to create
      lm:show_floors X A
    }

    V {
      #only one floor possible
      #Video: no window to create
      lm:show_floors X V
    }

    T {

      #Text: create and store window
      w_txt:create $channel $name
      w_fc:menu:tfloor_added $channel $name
    }
  }
}
proc fh:destroy_floor_win f {

  set type [fd:get_floortype $f]
  set channel [fd:channel $f]

    switch -glob -- $type {
```

```
    A {
      #only one floor possible
      lm:hide_floors X A
    }

    V {
      #only one floor possible
      lm:hide_floors X V
    }

    T {

      #Text: delete window
      w_fc:menu:tfloor_deleted $channel
      w_txt:destroy $channel
    }
  }



}
proc fh:set_message {dir f msg} {

  set t [fd:get_floortype $f]
  set c [fd:channel $f]

  switch -glob -- $t {

    A {
        if {$dir != "R"} {
          #write
          lm:ao:set_message "audio out $msg"
        }
        if {$dir != "W"} {
          #read
          lm:ai:set_message "audio in $msg"
        }
      }

    V {
        if {$dir != "R"} {
          #write
          lm:vo:set_message "video out $msg"
        }
        if {$dir != "W"} {
          #read
          lm:vi:set_message "video in $msg"
        }
      }

    T {
        if {$dir != "R"} {
          #write not permitted
          w_txt:out_message $c "Text write $msg"
        }
        if {$dir != "W"} {
          #read not permitted
          w_txt:in_message $c "Text read $msg"
        }
      }
   }
}
proc fh:set_init_message f {

  if {[fd:check_user_is_fc_for_floors 0 $f]} {
    #is controller ?
    #has token for his own floor
    fh:set_message X $f "controlled by you"
  } else {
    if {[fd:is_free $f]} {
      #free means: everybody has a token
      fh:set_message X $f "uncontrolled"
    } else {
      if {[fd:has_token W $f]} {
        fh:set_message W $f "ENABLED"
      } else {
        fh:set_message W $f "disabled"
      }
      if {[fd:has_token R $f]} {
        fh:set_message R $f "ENABLED"
      } else {
        fh:set_message R $f "disabled"
      }
    }
  }

  set dir [fd:controldir $f]

  if {$dir == "R"} {
    fh:set_message W $f "uncontrolled"
  }
  if {$dir == "W"} {
    fh:set_message R $f "uncontrolled"
  }
}
proc fh:add_floor {f fl_type_t fl_channel_t fl_name_t
fl_w_mute_t fl_r_mute_t} {

  main:status_message "Adding floor $f: $fl_name_t"

  fd:add_floor $f $fl_type_t $fl_channel_t $fl_name_t
$fl_r_mute_t $fl_w_mute_t

  #initialise floor-use-window
  fh:create_floor_win  $fl_type_t $fl_channel_t $fl_name_t
  w_fi:list:add $f
  fh:set_init_message $f
  w_fi:show_floor_info $f

}
proc fh:delete_floor f {

  main:status_message "Deleting floor $f"

  fh:destroy_floor_win $f
  w_fi:list:delete $f
  fd:delete_floor $f

}
proc fh:delete_all_floors {} {

  set fi [fd:all_floor_ids]

  foreach f $fi {
   catch {fh:delete_floor $f}
  }
  fd:reset_floors
}
proc fh:set_messages {media msg} {

  foreach m $media {

    if {$m != ""} {
      set dir [fd:dir $m]
      fh:set_message $dir [fd:id $m] $msg
     }
   }

}
proc fh:audio_unusable {} {

 #audio is inpossible, so make floors unusable
 foreach chn [fd:all_channels_for_floortype A] {
   set m [fd:id_from_channeltype $chn A]
   fd:set_unusable R $m
   fd:set_unusable W $m
   w_fi:show_floor_info $m
 }

 lm:hide_floors X A
}
proc fh:video_write_unusable {} {

 #video write is inpossible, so make floors unusable
 foreach chn [fd:all_channels_for_floortype V] {
   set m [fd:id_from_channeltype $chn V]
   fd:set_unusable W $m
   w_fi:show_floor_info $m
 }

  lm:hide_floors W V
}
proc fh:get_full_floormanes media {

  set mtext ""

   foreach f $media {
    set dir [fd:dir $f]
    if {$dir == "X"} {
      set dir "RW"
    }
    set mtext "$mtext + [fd:name [fd:id $f]]\($dir\)"
  }

  return [string range $mtext 3 end]
}
proc fh:video_read_unusable {} {

 #video read is inpossible, so make floors unusable
 foreach chn [fd:all_channels_for_floortype V] {
   set m [fd:id_from_channeltype $chn V]
   fd:set_unusable R $m
   w_fi:show_floor_info $m
 }

  lm:hide_floors R V
}
proc fh:set_unusable {dir f} {

  w_fi:show_floor_info $f
  lm:hide_floors $dir [fd:get_floortype $f]

}
#- TOP LEVEL procedures---------------------------------

set Name .main1
set Parent $Name
global Procs
set Procs($Name) { w_fc:color_darken w_fc:conv_singel_col
w_fc:stop_hci w_fc:kill_windows w_fc:show_listmenu
w_fc:stop_conference w_fc:snew_floorcontroller
w_fc:request_deleted w_fc:floors_grabbed w_fc:floors_granted
w_fc:mark_granted w_fc:grant_floors w_fc:return_floor
w_fc:grasp_floors w_fc:grab_floors w_fc:refuse_floor
w_fc:request_floor w_fc:add_request w_fc:withdraw_floor
w_fc:add_participant w_fc:delete_participant w_fc:set_listmenus
w_fc:first_initialisation w_fc:floors_returned
w_fc:floors_canceled w_fc:grasp_floors w_fc:floors_refused
w_fc:start w_fc:grab_floors_done w_fc:controller_status_change
w_fc:refuse_popup w_fc:show_faces w_fc:vidw_in_hide}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) w_fc:
set Glob_prefix($Name) w_fc_
proc w_fc:color_darken {color percent} {

  global w_fc_namelistbox

  set rgb [winfo rgb $w_fc_namelistbox $color]
  #set rgb [winfo rgb $color]

  return "[format "#%04x%04x%04x" [w_fc:conv_singel_col [lindex
$rgb 0] $percent] [w_fc:conv_singel_col [lindex $rgb 1]
$percent] [w_fc:conv_singel_col [lindex $rgb 2] $percent]]"
```

```
}
proc w_fc:conv_singel_col {col factor} {

  set ret [expr round ($col * $factor / 100)]

  if {$ret > 65535} {
    set ret 65535
  }

  return $ret
}
proc w_fc:stop_hci {} {

  global w_fc_stop_command

  #Clear all lists
  w_fc:rlist:clear
  rg:clear_reqlist
  w_fc:plist:clear
  ud:clear_meetinglist
  rg:clear_grantlist

  fh:delete_all_floors

  #show old startwindow
  eval {$w_fc_stop_command}

  w_fc:kill_windows

  #stop background processing
  sy:kill_afters

  main:status_message "Ready"

}
proc w_fc:kill_windows {} {

  global w_fc_use_pict

 #hide windows

  catch {windestroy .main1}

   if {$w_fc_use_pict} {
     ms_ihci:command destroy 0 0
   }

   #mock-ups
   catch {windestroy .whitboard}
   catch {windestroy .groupmgr}

  #popups
  catch {w_freq_hide}
  catch {w_fp:hide}
  w_fi:destroy

  lm:exit_win

  w_sc:exit_win
}
proc w_fc:show_listmenu {lbox x y} {

  update idletasks
  set ystart [winfo rooty "$lbox"]
  #set litem [$lbox curselection]
  set litem [$lbox nearest [expr $y - $ystart]]
  set coord [$lbox bbox $litem]
  set y [expr [lindex $coord 1] + [lindex $coord 3] + $ystart]
  tk_popup $lbox.m $x $y

}
proc w_fc:stop_conference {} {

  global w_fc_stop_pressed

  if {$w_fc_stop_pressed} {return}

  main:status_message "Stopping conference"

  set w_fc_stop_pressed 1
  nl:stop_netlayer
  w_fc:stop_hci

  #all is stopped
  main:status_message "READY"

}
proc w_fc:snew_floorcontroller {first media c_ipnr} {

  if {$first} {
   #first controller: request now possible
    w_fc:access:controller_known $media
    w_fc:menu:floor_controller_known

   #make sound
   bell
 }

  foreach f [fd:id $media] {
    fh:set_init_message $f
  }
  w_fi:show_floor_info $media
}
proc w_fc:request_deleted {m_left ipnr} {

  w_fc:rlist:delete $m_left $ipnr

  #normal entry in floorlist
  if {$m_left == ""} {
     w_fc:plist:set_props $ipnr "" black
  }
}
proc w_fc:floors_grabbed {real media text} {

  global w_fc_use_pict w_fc_my_name

  #make sound
  bell

  w_fc:menu:floor_return
  fh:set_messages $media "disabled"
  set m_left [rg:clear_granted_floors]
  w_fi:show_floor_info $media
  if {$w_fc_use_pict && ($m_left == "")} {
    # ms_ihci:command tokengrab $w_fc_my_name byname
     ms_ihci:untalkuserbyname $w_fc_my_name
  }

  w_fc:access:grabbed $real $media $text

}
proc w_fc:floors_granted media {

  global w_fc_use_pict w_fc_my_name

  fh:set_messages $media "ENABLED"
  rg:clear_requested_floors
  rg:add_granted_floors $media
  w_fc:access:granted $media
  w_fc:menu:floor_granted
  w_fi:show_floor_info $media

  #make sound
  bell

  #show change in picture, if enabled
  if {$w_fc_use_pict} {
    ms_ihci:talkuserbyname $w_fc_my_name
   #ms_ihci:command tokengive $w_fc_my_name byname
  }
}
proc w_fc:mark_granted {media ipnr} {

     set dark_green [w_fc:color_darken green 50]

  if {$media == ""} {
    w_fc:plist:set_props $ipnr "" black
  } else {
    #mark as granted in green. Req-list is optional
    w_fc:rlist:set_props $ipnr "* [rg:get_ud_request $ipnr]
\[[fh:get_full_floormanes $media]\]" $dark_green
    w_fc:plist:set_props $ipnr "* \[[fh:get_full_floormanes
$media]\]" $dark_green
  }
}
proc w_fc:grant_floors {media ipnr} {

  #just to be sure: control could have changed
  set media [fd:own_controlled_floors $media]

  global w_fc_use_pict

  if {[fc:grant_floor $media $ipnr]} {

    rg:add_to_grantlist $media $ipnr

    rg:delete_from_reqlist [rg:requested_media $ipnr] $ipnr
    w_fc:mark_granted $media $ipnr

    #show change in picture, if enabled
    if {$w_fc_use_pict} {
      #ms_ihci:command tokengive [ud:name_in_meetinglist $ipnr
0] byname
      ms_ihci:talkuserbyname [ud:name_in_meetinglist $ipnr 0]
    }

   #Show video, if enabled
   foreach f [fd:id $media] {
    if {[fd:get_floortype $f] == "V"} {
          set w_fc_vidr_ch [fd:channel $f]
        fu:add_video_read [fd:channel $f] $ipnr}
    }
  }
}
proc w_fc:return_floor media {

  global w_fc_use_pict w_fc_my_name

   if {[fc:return_floor $media]} {
      fh:set_messages $media "disabled"
      w_fc:access:returned $media
      w_fc:menu:floor_return
      set m_left [rg:clear_granted_floors]
      w_fi:show_floor_info $media

      if {$w_fc_use_pict && ($m_left == "")} {
        ms_ihci:untalkuserbyname $w_fc_my_name
        #ms_ihci:command tokenrelease $w_fc_my_name byname
      }
   }
}
proc w_fc:grasp_floors {} {

  global w_fc_use_pict

  fc:grasp_floor [fd:own_controlled_floors [fd:all_floor_ids]]

  #no floor left: user-picture can be changed
  if {$w_fc_use_pict} {
    #ms_ihci:command reset [ud:name_in_meetinglist $ipnr 0]
byname
    ms_ihci:reset
  }
```

```
}
proc w_fc:grab_floors {media ipnr} {

  if {[rg:is_in_grantlist $ipnr]} {
    if {[fc:grab_floor $media $ipnr]} {
      #status info
      main:status_message "Token being grabbed from $ipnr for
$media"

    } else {
    main:status_message "ERROR: Your are no controller for
$media !"
    }
  } else {
      #status info
      main:status_message "ERROR: $ipnr has no $media"
  }
}
proc w_fc:refuse_floor {media ipnr userdata} {

    global w_fc_use_pict

    #is floor  already given ?
    if {$media != ""} {
     if {[rg:granted_media $ipnr] == ""} {

      #make refusal known to the one requesting.
      if {[fc:refuse_floor $media $ipnr $userdata]} {

        #status info
        main:status_message "Floors $media refused for $ipnr"

        #request deleted
        set m_left [rg:delete_from_reqlist $media $ipnr]
        w_fc:request_deleted $m_left $ipnr

        if {$w_fc_use_pict} {
          #ms_ihci:command tokenreject [ud:name_in_meetinglist
$ipnr 0] byname
        }

      }

    } else {

      #floor cannot be refused if acces has been granted
      main:status_message "ERROR: Floor already given to $ipnr"
    }
   } else {
      main:status_message "ERROR: No floor requested by $ipnr"
   }
}
proc w_fc:request_floor {media userdata} {

  if {[fc:request_floor $media $userdata]} {
    rg:set_requested_floors $media
    w_fc:access:requested $media
    w_fc:menu:floor_requested
    w_fi:show_floor_info $media
  }
}
proc w_fc:add_request {media ipnr userdata} {

  global w_fc_use_pict

  set dark_red [w_fc:color_darken red 80]

  if {[ud:is_in_meetinglist $ipnr]} {

   #ip-number is added to the request list.
   rg:add_to_reqlist $media $ipnr $userdata

   if {$media == ""} {
     set color black
     set mi ""
   } else {
     set color $dark_red
     set mi "\[[fh:get_full_floormanes $media]\]"
   }

   #show request
   w_fc:rlist:add $ipnr
   #mark entry as "requesting" in both lists

   #put a ? behind it, with the question and
   w_fc:rlist:set_props $ipnr "? $userdata $mi" $color
   #put a ? behind it and mark as red
   w_fc:plist:set_props $ipnr "? $mi" $color

  if {$w_fc_use_pict} {
    #ms_ihci:command tokenplease [ud:name_in_meetinglist $ipnr
0] byname
  }

  } else {
   #problem !
   main:status_message "ERROR: request from Unknown ip-nr !!!"
  }
}
proc w_fc:withdraw_floor media {

  fc:withdraw_f_request $media
  w_fc:access:withdrawn $media
  w_fc:menu:floor_withdrawn
  rg:clear_requested_floors
  w_fi:show_floor_info $media

}
proc w_fc:add_participant {ipnr uid} {

  global w_fc_testmode w_fc_use_pict

  #someone joins the groupmeeting

   #convert to number, if needed
   set nr [ud:get_usernumber $uid]

   #now convert to complete name
   if {$nr >= 0} {
     set name [ud:get_username $nr 0]
   } else {
     set name $uid
   }

  #is this ip-number still in the classlist ?
   if {[ud:is_in_meetinglist $ipnr]} {

    #DO NOTHING: is probably network-error recovery....
    #first clear all entrys for that ip-number !

    # delete_from_classroom $ipnr
    # main:status_message "$ipnr was suddenly replaced !"

  } else {

    #now add the new meeting member to all lists
    ud:add_to_meetinglist $ipnr $name $nr
    w_fc:plist:add $ipnr
    w_fc:menu:user_added [ud:name_in_meetinglist $ipnr 1] $ipnr
    w_fc:plist:set_props $ipnr "" black

    #if enabled, show picture
    if {$w_fc_use_pict} {
     # ms_ihci:adding $name byname
      ms_ihci:adduserbyname $name
    }
  }

  #use database definition to find out about floorcontrol

  #check for controlled floors
  set mc [fr:check_for_controller $nr]

  #Ignore non-existing media
  set mc [fd:floors_exists $mc]

  if {$mc != ""} {
    fc:floor_controlled $mc $ipnr

  }

  if {$w_fc_testmode} {
    #testmode: use grab to announce control
    set media [fd:get_floors_controlled_by X]
    if {$media != ""} {
    fc:grab_floor $media $ipnr
    }
  }
}
proc w_fc:delete_participant ipnr {

  global w_fc_use_pict

  #someone leaves the tele-classroom

   if {[ud:is_in_meetinglist $ipnr]} {

    #delete from request list (if listed)
    rg:delete_from_reqlist [rg:requested_media $ipnr] $ipnr

    #delete from granted list (if listed)
    rg:delete_from_grantlist [rg:granted_media $ipnr] $ipnr

    #show this

    #if it was enabled, delete picture
    if {$w_fc_use_pict} {
      ms_ihci:deleteuserbyname [ud:name_in_meetinglist $ipnr 0]
      #ms_ihci:command deleting [ud:name_in_meetinglist $ipnr
0] byname
    }

    w_fc:request_deleted "" $ipnr
    w_fc:plist:delete $ipnr
    w_fc:menu:user_deleted $ipnr

    #delete from main lists
    ud:delete_from_meetinglist $ipnr

    fc:floors_not_controlled [fd:get_floors_controlled_by
$ipnr]

  } else {
    #oh oh !
    main:status_message "Unknown ip-nr !!!"
  }

}
proc w_fc:set_listmenus control {

  w_fc:plist:define_menu $control
  w_fc:rlist:define_menu $control

}
proc w_fc:first_initialisation {scommand do_debug} {

  global w_fc_b_floor_panic
  global w_fc_namelistbox w_fc_requestlistbox
  global w_fc_my_name w_fc_name_out
  global w_fc_stop_pressed
  global w_fc_stop_command
  global w_fc_testmode
  global w_fc_use_pict
```

```
  set w_fc_stop_pressed 0

   set w_fc_use_pict 0

   #mock-ups
   wb:init
   wincreate .groupmgr
   whide .groupmgr

  wincreate .main1

  set w_fc_stop_command $scommand

  set w_fc_testmode $do_debug

  set w_fc_b_floor_panic ".main1.b_panic"
  set w_fc_namelistbox ".main1.frame_class.fb_listb1"
  set w_fc_requestlistbox ".main1.frame_requests.fb_listb2"

  #intialise floor access indication frame
  w_fc:menu:init_menubar

  ud:clear_meetinglist
  w_fc:plist:clear
  rg:clear_grantlist
  rg:clear_reqlist
  w_fc:rlist:clear
  rg:clear_granted_floors
  rg:clear_requested_floors

  set w_fc_my_name "1"
  set w_fc_my_name [ud:get_current_username 0]
  set w_fc_name_out [ud:get_current_username 1]

  w_fc:access:init
  w_fi:init

  whide .main1
}
proc w_fc:floors_returned {media ipnr} {

   #message
   main:status_message "$ipnr returns floors $media"

   global w_fc_use_pict

    set m_left [rg:delete_from_grantlist $media $ipnr]

    #delete from requestlist
    w_fc:request_deleted $m_left $ipnr

    #stop showing videowindows
    #and prepare for future reception
    w_fc:vidw_in_hide $ipnr $media

    #no floor left: user-picture can be changed
    if {$w_fc_use_pict && ($m_left == "")} {
      #ms_ihci:command tokenrelease [ud:name_in_meetinglist
$ipnr 0] byname
      ms_ihci:untalkuserbyname [ud:name_in_meetinglist $ipnr 0]

   }
}
proc w_fc:floors_canceled {media ipnr} {

    global w_fc_use_pict

   #message
   main:status_message "$ipnr cancels request for $media"

   set m_left [rg:delete_from_reqlist $media $ipnr]
    #delete from requestlist
    w_fc:request_deleted $m_left $ipnr

   #already granted ?...mark as "granted without request"
   set m_gr [rg:granted_media $ipnr]
   if {$m_gr != ""} {
     w_fc:mark_granted $m_gr $ipnr
   }

   if {$w_fc_use_pict && ($m_left == "")} {
      #ms_ihci:command tokenwithdraw [ud:name_in_meetinglist
$ipnr 0] byname
   }
}
proc w_fc:floors_refused {media ipnr userdata} {

  global w_fc_use_pict

  #message
  main:status_message "$ipnr rejects flooracces for $media
because $userdata"

  #make sound
  bell

  w_fc:access:rejected $media $ipnr $userdata
  w_fc:menu:floor_withdrawn

  rg:clear_requested_floors
  w_fi:show_floor_info $media

   if {$w_fc_use_pict} {
      #ms_ihci:command tokenreject [ud:name_in_meetinglist
$ipnr 0] byname
   }

}
proc w_fc:start pict {

  global w_fc_my_name w_fc_actor_indication
  global w_fc_conf_id
  global w_fc_use_pict

  set w_fc_use_pict $pict
  if {$w_fc_use_pict} {
   #commands for old, working version
    #enable faces
    ms_ihci:init_image_hci 3
    #hide until requested
    ms_ihci:hideall
    #set picture
    #ms_ihci:adduserbyname $w_fc_my_name
    ms_ihci:adduserbyname [ud:get_current_username 0]
    #Commands for new, not working version.
    #enable faces
    #ms_ihci:command init [fr:current_mnr] 3
    #hide until requested
    #ms_ihci:command hide 0 0
    #set picture
    #ms_ihci:command adding $w_fc_my_name byname
    w_fc:controller_status_change ""
  }

  w_fc:access:controller_unknown
  w_fc:menu:floor_controller_unknown

  wshow .main1

  set w_fc_conf_id [fr:current_mname]
  set w_fc_actor_indication [fr:get_actor_indication]

  focus .main1

}
proc w_fc:grab_floors_done {media ipnr} {

  global w_fc_use_pict

  if {[rg:is_in_grantlist $ipnr]} {
    set result 1
    #any media left ?
    set m_left [rg:delete_from_grantlist $media $ipnr]
    w_fc:request_deleted $m_left $ipnr

    #stop showing videowindows
    #and prepare for future reception
    w_fc:vidw_in_hide $ipnr $media

    #no floor left: user-picture can be changed
    if {$w_fc_use_pict && ($m_left == "")} {
      #ms_ihci:command tokengrab [ud:name_in_meetinglist $ipnr
0] byname
      ms_ihci:untalkuserbyname [ud:name_in_meetinglist $ipnr 0]
  }

  } else {
    set result 0
    #status info
    w_sc:addlst_debug "ERROR: $ipnr had no $media"
  }

  return $result
}
proc w_fc:controller_status_change media {

  global w_fc_b_floor_panic w_fc_use_pict w_fc_my_name

  if {[fd:not_own_controlled_floors [fd:all_floor_ids]] == ""}
{
    #was last controller: request now impossible
    w_fc:access:controller_unknown
    w_freq_hide
  }

  if {[fd:get_floors_controlled_by X] == ""} {
    #no floors controlled
    $w_fc_b_floor_panic configure -state disabled
    w_fc:menu:is_controller 0
    w_fc:set_listmenus 0
    #show change in picture, if enabled

  } else {
    $w_fc_b_floor_panic configure -state normal
    w_fc:menu:is_controller 1
    w_fc:set_listmenus 1

  }

  w_fi:show_floor_info $media

  foreach f [fd:id $media] {
    fh:set_init_message $f
  }

}
proc w_fc:refuse_popup {media ipnr} {


    #is floor  already given ?
    if {$media != ""} {
     if {[rg:granted_media $ipnr] == ""} {
       # show popup-window
       w_fp:init $media $ipnr
       w_fp:show
     } else {
      #floor cannot be refused if acces has been granted
      main:status_message "ERROR: Floor already given to $ipnr"
     }
    } else {
      main:status_message "ERROR: No floor requested by $ipnr"
    }
}
proc w_fc:show_faces {} {
```

```
  global w_fc_use_pict

  if {$w_fc_use_pict} {
    ms_ihci:command show 0 0}
}
proc w_fc:vidw_in_hide {ipnr media} {

  #stop showing videowindows
  #and prepare for future reception

  global w_fc_vidr_ch w_fc_vidr_ipnr

  foreach f [fd:id $media] {
    if {[fd:get_floortype $f] == "V"} {
      set real [fu:delete_video_read [fd:channel $f] $ipnr]
      if {$real} {
        #wait a short time for the videostream to stop
        #because we only want a future videostream
          #Globalen zijn nodig omdat aan "after" geen
          #normale variabelen doorgegeven kunnen worden.
          set w_fc_vidr_ch [fd:channel $f]
          set w_fc_vidr_ipnr $ipnr
        after 1500 {fu:add_video_read $w_fc_vidr_ch
$w_fc_vidr_ipnr}
      }
    }
  }
}

set Name $Parent.l30
# Procedures----------------------------

set Name $Parent.l31
# Procedures----------------------------

set Name $Parent.b35
# Procedures----------------------------
global Procs
set Procs($Name) { {bind .main1.b35 <KeyPress-Return>}}

set Name $Parent.frame_requests
# Procedures----------------------------

set Name $Parent.frame_requests.fb_listb2
# Procedures----------------------------
global Procs
set Procs($Name) { w_fc:rlist:selection_grant
w_fc:rlist:selection_grab w_fc:rlist:selection_refuse
w_fc:rlist:selection_volume w_fc:rlist:selection_refuse_arg
{bind .main1.frame_requests.fb_listb2 <ButtonPress-1>}
w_fc:rlist:add w_fc:rlist:clear w_fc:rlist:delete
w_fc:rlist:set_props w_fc:rlist:define_menu {bind
.main1.frame_requests.fb_listb2 <KeyPress-Return>} {bind
.main1.frame_requests.fb_listb2 <KeyPress-space>}
w_fc:rlist:get_ipnr_selection}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) w_fc:rlist:
set Glob_prefix($Name) w_fc_rlist_
proc w_fc:rlist:selection_grant {} {

  set ipnr [w_fc:rlist:get_ipnr_selection]
  if {$ipnr != ""} {
    set media [rg:requested_media $ipnr]
    w_fc:grant_floors $media $ipnr
  }
}
proc w_fc:rlist:selection_grab {} {

  set ipnr [w_fc:rlist:get_ipnr_selection]
  if {$ipnr != ""} {
    w_fc:grab_floors [rg:granted_media $ipnr] $ipnr
  }
}
proc w_fc:rlist:selection_refuse {} {

  set ipnr [w_fc:rlist:get_ipnr_selection]
  if {$ipnr != ""} {
    w_fc:refuse_floor [rg:requested_media $ipnr] $ipnr ""
  }
}
proc w_fc:rlist:selection_volume vol {

  set ipnr [w_fc:rlist:get_ipnr_selection]
  if {$ipnr != ""} {
    foreach ch [fd:all_channels_for_floortype A] {
       fu:set_audiofloor_volume $ch $ipnr $vol
    }
  }
}
proc w_fc:rlist:selection_refuse_arg {} {

  set ipnr [w_fc:rlist:get_ipnr_selection]
  if {$ipnr != ""} {
    w_fc:refuse_popup [rg:requested_media $ipnr] $ipnr
  }
}
proc w_fc:rlist:add ipnr {

global w_fc_reqlist w_fc_requestlistbox

  set lnr [lsearch -glob $w_fc_reqlist $ipnr]
  if {$lnr < 0} {
    #no request present: append to list.
    lappend w_fc_reqlist $ipnr
    #insert empty item
    #use mark to set values proerly
    $w_fc_requestlistbox insert end " "
  } else {
    #double request: just ignore for now.
  }
}

proc w_fc:rlist:clear {} {

  global w_fc_requestlistbox w_fc_reqlist

  catch {$w_fc_requestlistbox delete 0 [expr [llength
$w_fc_reqlist] - 1]}
  set w_fc_reqlist ""

}
proc w_fc:rlist:delete {media ipnr} {

  global w_fc_reqlist w_fc_requestlistbox

  #find request
  set lnr [lsearch -glob $w_fc_reqlist $ipnr]
  if {$lnr >= 0} {

     #delete from request list
     set w_fc_reqlist [lreplace $w_fc_reqlist $lnr $lnr]
     $w_fc_requestlistbox delete $lnr

  } else {

    #ignore error for now
  }

}
proc w_fc:rlist:set_props {ipnr marksign color} {

  global w_fc_reqlist w_fc_requestlistbox w_fc_testmode

  #change the marking character in the request-listbox
  #behind the name, indicated by ipnr to marksign.

  #find position of request
  set lnr [lsearch -glob $w_fc_reqlist $ipnr]

  #is entry present ?
  if {$lnr >= 0} {

    #change contents
    #this must be done bij deleting & inserting.
    $w_fc_requestlistbox delete $lnr
    if {$w_fc_testmode} {
      $w_fc_requestlistbox insert $lnr "$ipnr
[ud:name_in_meetinglist $ipnr 1] $marksign"
    } else {
      $w_fc_requestlistbox insert $lnr "[ud:name_in_meetinglist
$ipnr 1] $marksign"
    }
    #change color
    $w_fc_requestlistbox item configure $lnr -foreground $color
  }

}
proc w_fc:rlist:define_menu control {

  global w_fc_requestlistbox

  set parent $w_fc_requestlistbox

  #create menu definition

  set menu ""
  if {$control} {
    lappend menu {{Enable speach} command
w_fc:rlist:selection_grant}
    lappend menu {{disable speach} command
w_fc:rlist:selection_grab}
    lappend menu {{Refuse Request} command
w_fc:rlist:selection_refuse}
    lappend menu {{Refuse with message} command
w_fc:rlist:selection_refuse_arg}
  }

  if {$menu != ""} {
    lappend menu {separator}
  }
  lappend menu {{Loud Volume} command
{w_fc:rlist:selection_volume 2}}
  lappend menu {{Normal Volume} command
{w_fc:rlist:selection_volume 1}}
  lappend menu {{Soft Volume} command
{w_fc:rlist:selection_volume 0.5}}

  set Menu_string($parent) "\{$menu\} -tearoff 0"

 #make menu itself
 eval "make_menu $parent $Menu_string($parent)"
}
proc w_fc:rlist:get_ipnr_selection {} {

  global w_fc_requestlistbox w_fc_reqlist

  if {[llength [$w_fc_requestlistbox curselection]] > 0} {
    set ipnr [lindex $w_fc_reqlist [$w_fc_requestlistbox
curselection]]
  } else {
    set ipnr ""
  }
  return $ipnr
}

set Name $Parent.frame_requests.l48
# Procedures----------------------------

set Name $Parent.frame_requests.sb92
# Procedures----------------------------

set Name $Parent.frame_class
# Procedures----------------------------
```

```
set Name $Parent.frame_class.fb_listb1
# Procedures----------------------------
global Procs
set Procs($Name) { w_fc:plist:selection_grant
w_fc:plist:selection_grab w_fc:plist:selection_refuse
w_fc:plist:selection_volume w_fc:plist:selection_vid_start
w_fc:plist:selection_vid_stop {bind
.main1.frame_class.fb_listb1 <ButtonPress-1>} w_fc:plist:add
w_fc:plist:delete w_fc:plist:set_props w_fc:plist:clear
w_fc:plist:define_menu {bind .main1.frame_class.fb_listb1
<KeyPress-Return>} {bind .main1.frame_class.fb_listb1
<KeyPress-space>} w_fc:plist:get_ipnr_selection
w_fc:plist:selection_refuse_arg}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) w_fc:plist
set Glob_prefix($Name) w_fc_plist_
proc w_fc:plist:selection_grant {} {

  set ipnr [w_fc:plist:get_ipnr_selection]
  if {$ipnr != ""} {
    set media  [rg:req_or_all_media $ipnr]
    w_fc:grant_floors $media $ipnr
  }
}
proc w_fc:plist:selection_grab {} {

  set ipnr [w_fc:plist:get_ipnr_selection]
  if {$ipnr != ""} {
    w_fc:grab_floors [rg:granted_media $ipnr] $ipnr
  }
}
proc w_fc:plist:selection_refuse {} {

  set ipnr [w_fc:plist:get_ipnr_selection]
  if {$ipnr != ""} {
    set media  [rg:requested_media $ipnr]
    w_fc:refuse_floor $media $ipnr ""
  }
}
proc w_fc:plist:selection_volume vol {

  set ipnr [w_fc:plist:get_ipnr_selection]
  if {$ipnr != ""} {
    foreach ch [fd:all_channels_for_floortype A] {
       fu:set_audiofloor_volume $ch $ipnr $vol
    }
  }
}
proc w_fc:plist:selection_vid_start {} {

  set ipnr [w_fc:plist:get_ipnr_selection]
  if {$ipnr != ""} {
    foreach ch [fd:all_channels_for_floortype V] {
      fu:add_video_read $ch $ipnr
    }
  }
}
proc w_fc:plist:selection_vid_stop {} {

  set ipnr [w_fc:plist:get_ipnr_selection]
  if {$ipnr != ""} {
    foreach ch [fd:all_channels_for_floortype V] {
       fu:delete_video_read $ch $ipnr
    }
  }
}
proc w_fc:plist:add ipnr {

  global w_fc_namelistbox w_fc_floorlist

  #Add member to the floorlist
  lappend w_fc_floorlist $ipnr

  #empty line: use w_fc:plist:set_props to fill this in.
  $w_fc_namelistbox insert end " "
}
proc w_fc:plist:delete ipnr {

  global w_fc_floorlist w_fc_namelistbox

  set lnr [lsearch -glob $w_fc_floorlist $ipnr]
  if {$lnr >= 0 } {
    set w_fc_floorlist [lreplace $w_fc_floorlist $lnr $lnr]
    $w_fc_namelistbox delete $lnr
  }
}
proc w_fc:plist:set_props {ipnr marksign color} {

  global w_fc_floorlist w_fc_namelistbox w_fc_testmode

  #this must be done bij deleting & inserting.
  set lnr [lsearch -glob $w_fc_floorlist $ipnr]
  if {$lnr >= 0} {
    $w_fc_namelistbox delete $lnr
    if {$w_fc_testmode} {
     $w_fc_namelistbox insert $lnr "$ipnr
[ud:name_in_meetinglist $ipnr 1] $marksign"
   } else {
     $w_fc_namelistbox insert $lnr "[ud:name_in_meetinglist
$ipnr 1] $marksign"
   }
   #change color
   $w_fc_namelistbox item configure $lnr -foreground $color

  } else {
    set er_message "Unknown ip-nr !!!"
  }

 }
proc w_fc:plist:clear {} {

  global w_fc_namelistbox w_fc_floorlist

  catch {$w_fc_namelistbox delete 0 [llength $w_fc_floorlist]}
  set w_fc_floorlist ""
}
proc w_fc:plist:define_menu control {

  global w_fc_namelistbox

  set parent $w_fc_namelistbox

  #this procedure is to slow to invoke when the menu is
  #called, so it must be predefined.

  #create menu definition

  set menu ""

  if {$control} {
    lappend menu {{Enable speach} command
w_fc:plist:selection_grant}
    lappend menu {{disable speach} command
w_fc:plist:selection_grab}
    lappend menu {{Refuse Request} command
w_fc:plist:selection_refuse}
    lappend menu {{Refuse with message} command
w_fc:plist:selection_refuse_arg}
  }

  if {$menu != ""} {
    lappend menu {separator}
  }
  lappend menu {{Loud Volume} command
{w_fc:plist:selection_volume 2}}
  lappend menu {{Normal Volume} command
{w_fc:plist:selection_volume 1}}
  lappend menu {{Soft Volume} command
{w_fc:plist:selection_volume 0.5}}
 lappend menu {{Zero Volume} command
{w_fc:plist:selection_volume 0}}


  if {$menu != ""} {
    lappend menu {separator}
  }
  lappend menu {{Receive video} command
{w_fc:plist:selection_vid_start}}
  lappend menu {{Stop video} command
{w_fc:plist:selection_vid_stop}}
    set Menu_string($parent) "\{$menu\} -tearoff 0"

 #make menu itself
 eval "make_menu $parent $Menu_string($parent)"
}
proc w_fc:plist:get_ipnr_selection {} {

  global w_fc_namelistbox w_fc_floorlist

  if {[llength [$w_fc_namelistbox curselection]] > 0 } {
    set ipnr [lindex $w_fc_floorlist [$w_fc_namelistbox
curselection]]
  } else {
    set ipnr ""
  }
  return $ipnr
}
proc w_fc:plist:selection_refuse_arg {} {

  set ipnr [w_fc:plist:get_ipnr_selection]
  if {$ipnr != ""} {
    set media  [rg:requested_media $ipnr]
    w_fc:refuse_popup $media $ipnr
  }
}

set Name $Parent.frame_class.l54
# Procedures----------------------------

set Name $Parent.frame_class.sb89
# Procedures----------------------------

set Name $Parent.floor
# Procedures----------------------------
global Procs
set Procs($Name) { w_fc:access:set_tlight w_fc:access:set_image
w_fc:access:requested w_fc:access:withdrawn
w_fc:access:returned w_fc:access:controller_known
w_fc:access:grabbed w_fc:access:rejected w_fc:access:granted
w_fc:access:init w_fc:access:controller_unknown
w_fc:access:change_trlight w_fc:access:return
w_fc:access:withdraw}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) w_fc:access:
set Glob_prefix($Name) w_fc_access_
proc w_fc:access:set_tlight color {

  global fl_traffic_sign

  if {[fd:get_floors_controlled_by "X"] == ""} {
    set ext ""
  } else {
    set ext "2"
  }

    #image (traffic light)
    set cmd "$fl_traffic_sign configure -image
light_$color$ext"
    eval $cmd
}
proc w_fc:access:set_image img {

 global fl_traffic_sign
```

```
  #image (normally traffic light)
  $fl_traffic_sign configure -image $img

}
proc w_fc:access:requested media {

  global w_fc_fl_breq w_fc_fl_bwithdraw

  #set button
  $w_fc_fl_breq configure -relief sunken
  $w_fc_fl_breq configure -state disabled
  $w_fc_fl_bwithdraw configure -state normal

  w_fc:access:set_tlight "yellow"

  #status info
  set er_message "Floor requested"

}
proc w_fc:access:withdrawn media {

  global w_fc_fl_breq w_fc_fl_bwithdraw

    #status info
    main:status_message "Request for token withdrawn"

    #show button
    $w_fc_fl_breq configure -relief raised
    $w_fc_fl_bwithdraw configure -state disabled
    $w_fc_fl_breq configure -state normal

    #show image (traffic light)
    w_fc:access:set_tlight "red"
}
proc w_fc:access:returned media {

global w_fc_fl_breq w_fc_fl_breturn

  #status info
  main:status_message "Floor returned"

  w_fc:access:change_trlight

}
proc w_fc:access:controller_known media {

  global w_fc_fl_breq

  #signal
  w_fc:access:set_tlight "red"
  $w_fc_fl_breq configure -state active
}
proc w_fc:access:grabbed {real media text} {

global w_fc_fl_breq w_fc_fl_breturn

  #controller cannot be grabbed !
  if {[fd:check_user_is_fc_for_floors 0 [fd:id $media]]}
{return}

  w_fc:access:set_image "stop"

  if {$real} {

    #give clear message
    tk_dialog ".tkdialog" "Floor Control" "$text for \n
[fh:get_full_floormanes $media]" "" 0 "OK"

  }
  w_fc:access:change_trlight
}
proc w_fc:access:rejected {media ipnr userdata} {

  global w_fc_fl_breq w_fc_fl_bwithdraw

  #do nothing yet

  #set button
  $w_fc_fl_breq configure -relief raised
  $w_fc_fl_bwithdraw configure -state disabled
  $w_fc_fl_breq configure -state normal

  w_fc:access:set_image "stop"

   #give clear message
    tk_dialog ".tkdialog" "Floor control" "Floor refused for
\n[fh:get_full_floormanes $media]\n\n$userdata" "" 0 "OK"

  #image (traffic light)
  w_fc:access:set_tlight "red"


}
proc w_fc:access:granted media {

  global w_fc_fl_breturn w_fc_fl_breq w_fc_fl_bwithdraw

  #set button
  $w_fc_fl_breq configure -relief raised
  $w_fc_fl_breq configure -state disabled
  $w_fc_fl_bwithdraw configure -state disabled

  #set return-button
  $w_fc_fl_breturn configure -state normal

  #show image (traffic light)
  w_fc:access:set_tlight "green"
}
proc w_fc:access:init {} {

  global w_fc_fl_breq w_fc_fl_breturn w_fc_fl_bwithdraw
fl_traffic_sign

  set w_fc_fl_breq ".main1.floor.b_floor_req"
  set w_fc_fl_breturn ".main1.floor.b_return"
  set w_fc_fl_bwithdraw ".main1.floor.b_withdraw"
  set fl_traffic_sign ".main1.floor.l_image1"

  #request possible
  $w_fc_fl_breq configure -state normal

  #No return and witdraw possible yet.
  $w_fc_fl_breturn configure -state disabled
  $w_fc_fl_bwithdraw configure -state disabled
}
proc w_fc:access:controller_unknown {} {

   global w_fc_fl_breq

   #There is no floor left to ask for...
   #So, there are no floor-requests possible

   $w_fc_fl_breq configure -state disabled

   #user is self a controller?
    if {[fd:get_floors_controlled_by "X"] == ""} {
      w_fc:access:set_image not_allowed
    } else {
      w_fc:access:set_image light_green
    }
}
proc w_fc:access:change_trlight {} {

  global w_fc_fl_breturn w_fc_fl_breq w_fc_fl_bwithdraw

    #show image (traffic light)
    w_fc:access:set_tlight "red"
    update idletasks

    if {[fd:has_tokens [fd:not_own_controlled_floors
[fd:all_floor_ids]]]} {
      #still a token left. Just flash light a bit
      w_fc:access:set_tlight "green"

    } else {
      #change buttons
      $w_fc_fl_breq configure -state normal
      $w_fc_fl_breturn configure -state disabled
    }
}
proc w_fc:access:return {} {

  w_fc:return_floor [rg:get_granted_floors]

}
proc w_fc:access:withdraw {} {

w_fc:withdraw_floor [rg:get_requested_floors]

}

set Name $Parent.floor.l79
# Procedures----------------------------

set Name $Parent.floor.b_withdraw
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .main1.floor.b_withdraw <KeyPress-
Return>}}

set Name $Parent.floor.b_return
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .main1.floor.b_return <KeyPress-
Return>}}

set Name $Parent.floor.b_floor_req
# Procedures----------------------------
global Procs
set Procs($Name) { {bind .main1.floor.b_floor_req <KeyPress-
Return>}}

set Name $Parent.floor.l_image1
# Procedures----------------------------
global Procs
set Procs($Name) { {bind .main1.floor.l_image1 <KeyPress-
Return>}}

set Name $Parent.b_panic
# Procedures----------------------------
global Procs
set Procs($Name) { {bind .main1.b_panic <KeyPress-Return>}}

set Name $Parent.b121
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .main1.b121 <KeyPress-Return>}}

set Name $Parent.b200
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .main1.b200 <KeyPress-Return>}}

set Name $Parent.b184
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .main1.b184 <KeyPress-Return>}}

set Name $Parent.b185
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .main1.b185 <KeyPress-Return>}}

set Name $Parent.f_info
```

```
# Procedures----------------------------

set Name $Parent.f_info.l193
# Procedures----------------------------

set Name $Parent.f_info.l36
# Procedures----------------------------

set Name $Parent.f_info.l65
# Procedures----------------------------

set Name $Parent.f_info.l37
# Procedures----------------------------

set Name $Parent.f_info.l38
# Procedures----------------------------

set Name $Parent.fmenu
# Procedures----------------------------
global Procs
set Procs($Name) {w_fc:menu:floor_requested
w_fc:menu:floor_return w_fc:menu:floor_granted
w_fc:menu:floor_controller_unknown
w_fc:menu:floor_controller_known w_fc:menu:init_menubar
w_fc:menu:floor_withdrawn w_fc:menu:is_controller
w_fc:menu:user_added w_fc:menu:user_deleted
w_fc:menu:tfloor_added w_fc:menu:tfloor_deleted}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) w_fc:menu:
set Glob_prefix($Name) w_fc_menu_
proc w_fc:menu:floor_requested {} {

  global w_fc_menu_req w_fc_menu_withdr w_fc_menu_return

  .main1.fmenu.m_fa.m entryconfigure $w_fc_menu_req -state
disabled
  .main1.fmenu.m_fa.m entryconfigure $w_fc_menu_withdr -state
normal

}
proc w_fc:menu:floor_return {} {

  global w_fc_menu_req w_fc_menu_withdr w_fc_menu_return

    if {![fd:has_tokens [fd:not_own_controlled_floors
[fd:all_floor_ids]]]} {
     #has no token left.
    .main1.fmenu.m_fa.m entryconfigure $w_fc_menu_req -state
normal
    .main1.fmenu.m_fa.m entryconfigure $w_fc_menu_withdr -state
disabled
    .main1.fmenu.m_fa.m entryconfigure $w_fc_menu_return -state
disabled
  }
}
proc w_fc:menu:floor_granted {} {

  global w_fc_menu_req w_fc_menu_withdr w_fc_menu_return

  .main1.fmenu.m_fa.m entryconfigure $w_fc_menu_req -state
disabled
  .main1.fmenu.m_fa.m entryconfigure $w_fc_menu_withdr -state
disabled
  .main1.fmenu.m_fa.m entryconfigure $w_fc_menu_return -state
normal

}
proc w_fc:menu:floor_controller_unknown {} {

  global w_fc_menu_req w_fc_menu_withdr w_fc_menu_return

  .main1.fmenu.m_fa.m entryconfigure $w_fc_menu_req -state
disabled
  .main1.fmenu.m_fa.m entryconfigure $w_fc_menu_withdr -state
disabled
  .main1.fmenu.m_fa.m entryconfigure $w_fc_menu_return -state
disabled

}
proc w_fc:menu:floor_controller_known {} {

  global w_fc_menu_req w_fc_menu_withdr w_fc_menu_return

  .main1.fmenu.m_fa.m entryconfigure $w_fc_menu_req -state
normal

}
proc w_fc:menu:init_menubar {} {

  global w_fc_menu_req w_fc_menu_withdr w_fc_menu_return
  global w_fc_menu_panic w_fc_menu_e_spc w_fc_menu_d_spc
w_fc_menu_r_spc w_fc_menu_index_usr
  global w_fc_menu_txt w_fc_menu_index_t_ch

  set w_fc_menu_req 0
  set w_fc_menu_withdr 1
  set w_fc_menu_return 2

  set w_fc_menu_panic 0
  set w_fc_menu_e_spc 1
  set w_fc_menu_d_spc 2
  set w_fc_menu_r_spc 3

  set w_fc_menu_txt 1
  catch {unset w_fc_menu_index_t_ch}
  catch {unset w_fc_menu_index_usr}

  .main1.fmenu.m_fc.m entryconfigure $w_fc_menu_e_spc -
underline 0
  .main1.fmenu.m_fc.m entryconfigure $w_fc_menu_d_spc -
underline 0
  .main1.fmenu.m_fc.m entryconfigure $w_fc_menu_r_spc -
underline 0
  .main1.fmenu.m_w.m entryconfigure $w_fc_menu_txt -underline 0

}
proc w_fc:menu:floor_withdrawn {} {

  global w_fc_menu_req w_fc_menu_withdr w_fc_menu_return

  .main1.fmenu.m_fa.m entryconfigure $w_fc_menu_req -state
normal
  .main1.fmenu.m_fa.m entryconfigure $w_fc_menu_withdr -state
disabled

}
proc w_fc:menu:is_controller c {

  global w_fc_menu_panic w_fc_menu_e_spc w_fc_menu_d_spc
w_fc_menu_r_spc

  if {$c} {
     .main1.fmenu.m_fc.m entryconfigure $w_fc_menu_panic -state
normal
     .main1.fmenu.m_fc.m entryconfigure $w_fc_menu_e_spc -state
normal
     .main1.fmenu.m_fc.m entryconfigure $w_fc_menu_d_spc -state
normal
     .main1.fmenu.m_fc.m entryconfigure $w_fc_menu_r_spc -state
normal
  } else {
     .main1.fmenu.m_fc.m entryconfigure $w_fc_menu_panic -state
disabled
     .main1.fmenu.m_fc.m entryconfigure $w_fc_menu_e_spc -state
disabled
     .main1.fmenu.m_fc.m entryconfigure $w_fc_menu_d_spc -state
disabled
     .main1.fmenu.m_fc.m entryconfigure $w_fc_menu_r_spc -state
disabled
  }
}
proc w_fc:menu:user_added {name ipnr} {

  global w_fc_menu_panic w_fc_menu_e_spc w_fc_menu_d_spc
w_fc_menu_r_spc w_fc_menu_index_usr

  set win ".main1.fmenu.m_fc.m.m"

  #set enable menu
  set smenu $win[.main1.fmenu.m_fc.m index $w_fc_menu_e_spc]
  set CMD "$smenu add command -label \"$name\" -command
\{w_fc:grant_floors \"\[rg:req_or_all_media $ipnr\]\" $ipnr\}"
  eval $CMD

  #set disable menu
  set smenu $win[.main1.fmenu.m_fc.m index $w_fc_menu_d_spc]
  set CMD "$smenu add command -label \"$name\" -command
\{w_fc:grab_floors \"\[rg:granted_media $ipnr\]\" $ipnr\}"
  eval $CMD

  #set refuse menu
  set smenu $win[.main1.fmenu.m_fc.m index $w_fc_menu_r_spc]
  set CMD "$smenu add command -label \"$name\" -command
\{w_fc:refuse_floor \"\[rg:requested_media $ipnr\]\" $ipnr
\"\"\}"
  eval $CMD

  #store menu-index for user
  set w_fc_menu_index_usr($ipnr) [$smenu index last]
}
proc w_fc:menu:user_deleted ipnr {

  global w_fc_menu_panic w_fc_menu_e_spc w_fc_menu_d_spc
w_fc_menu_r_spc w_fc_menu_index_usr

  set win ".main1.fmenu.m_fc.m.m"

  set i $w_fc_menu_index_usr($ipnr)

  set smenu $win[.main1.fmenu.m_fc.m index $w_fc_menu_e_spc]
  set CMD "$smenu delete \"$i\""
  eval $CMD

  set smenu $win[.main1.fmenu.m_fc.m index $w_fc_menu_d_spc]
  set CMD "$smenu delete \"$i\""
  eval $CMD

  set smenu $win[.main1.fmenu.m_fc.m index $w_fc_menu_r_spc]
  set CMD "$smenu delete \"$i\""
  eval $CMD
}
proc w_fc:menu:tfloor_added {channel name} {

  global w_fc_menu_txt w_fc_menu_index_t_ch

  set win ".main1.fmenu.m_w.m.m"

  #add textfloor to menu
  set smenu $win[.main1.fmenu.m_w.m index $w_fc_menu_txt]
  set CMD "$smenu add command -label \"$name\" -command
\"w_txt:show $channel\""
  eval $CMD

  #store index of textfloor menu-item
  set w_fc_menu_index_t_ch($channel) [$smenu index last]
}
proc w_fc:menu:tfloor_deleted channel {

  global w_fc_menu_txt w_fc_menu_index_t_ch

  set win ".main1.fmenu.m_w.m.m"
  set smenu $win[.main1.fmenu.m_w.m index $w_fc_menu_txt]
```

```
  set CMD "$smenu delete \"$w_fc_menu_index_t_ch($channel)\""
  eval $CMD

}

set Name $Parent.fmenu.m_fc
# Procedures----------------------------

set Name $Parent.fmenu.m_w
# Procedures----------------------------

set Name $Parent.fmenu.m_f
# Procedures----------------------------

set Name $Parent.fmenu.m_fa
# Procedures----------------------------

set Name $Parent.fmenu.m203
# Procedures----------------------------

set Name $Parent.b201
# Procedures----------------------------
#- TOP LEVEL procedures--------------------------------

set Name .media1
set Parent $Name
global Procs
set Procs($Name) { {bind .media1 <Alt-h>}}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) lm:
set Glob_prefix($Name) lm_

set Name $Parent.f_vid_in
# Procedures----------------------------
global Procs
set Procs($Name) { lm:vi:enable lm:vi:toggle_frame
lm:vi:set_message}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) lm:vi:
set Glob_prefix($Name) lm_vi_
proc lm:vi:enable {} {

  global lm_vi_bmute

    if {$lm_vi_bmute} {
      foreach ch [fd:all_channels_for_floortype V] {
        set f [fd:id_from_channeltype $ch V]
        fu:set_mute R $f
      }
    } else {
    set succes 0
    foreach ch [fd:all_channels_for_floortype V] {
      set f [fd:id_from_channeltype $ch V]
      set succes [expr [fu:clear_mute R $f] || $succes]
    }
    set lm_vi_bmute [expr !$succes]
  }
}
proc lm:vi:toggle_frame {} {

  global lm_shrunk_vid_i lm_frame_vid_i

  if {$lm_shrunk_vid_i} {
    lm:frame_expand $lm_frame_vid_i
    set lm_shrunk_vid_i 0
  } else {
    lm:frame_shrink $lm_frame_vid_i
    set lm_shrunk_vid_i 1
  }
}
proc lm:vi:set_message text {

  global lm_vi_msg

  set lm_vi_msg $text

}

set Name $Parent.f_vid_in.l142
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .media1.f_vid_in.l142 <ButtonPress-1>}}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) lm:vi:l142:
set Glob_prefix($Name) lm_vi_l142_

set Name $Parent.f_vid_in.b143
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.f_vid_in.l_msg
# Procedures----------------------------

set Name $Parent.f_vid_out
# Procedures----------------------------
global Procs
set Procs($Name) {lm:vo:enable lm:vo:change lm:vo:set_vsize
lm:vo:toggle_frame lm:vo:set_message lm:vo:disable
lm:vo:set_vsizebar}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) lm:vo:
set Glob_prefix($Name) lm_vo_
proc lm:vo:enable {} {

  #set the defined parameters for video-output,
  #display the actual returned size
  #and start video-out

  global lm_vo_bmute lm_vo_size lm_vo_preview

  #set parameters

  lm:vo:set_vsizebar [nl:prepare_vid_out_props $lm_vo_size
$lm_vo_preview]

  if {!$lm_vo_bmute} {
    set succes 0
    foreach ch [fd:all_channels_for_floortype V] {
      set f [fd:id_from_channeltype $ch V]
      set succes [expr [fu:clear_mute W $f] || $succes]
    }
    set lm_vo_bmute [expr !$succes]
  }
}
proc lm:vo:change {} {

  global lm_vo_change

  #change something to video write. First collect changes
  #for a small period of time. After that, change video
  #write. When this is called in the middle of a change,
  #wait for first change to complete becuase new settings
  #can be taken into account anyway.

  if {![sy:stophci]} {
    #once is enough
    if {!$lm_vo_change} {
      set lm_vo_change 1

      w_sc:addlst_debug "video out parameter change"

      #no more calls during wait..
      #time-out to collect changes from HCI

      after 2000 {

        #now change !
        lm:vo:disable
        lm:vo:enable

        #another change is possible
        set lm_vo_change 0
      }
    }
  }
}
proc lm:vo:set_vsize dummy {

  global lm_mstart

  if {[sy:stophci]} {return}
  if {$lm_mstart} {return}

  after 100 {lm:vo:change}


}
proc lm:vo:toggle_frame {} {

  global lm_shrunk_vid_o lm_frame_vid_o

  if {$lm_shrunk_vid_o} {
    lm:frame_expand $lm_frame_vid_o
    set lm_shrunk_vid_o 0
  } else {
    lm:frame_shrink $lm_frame_vid_o
    set lm_shrunk_vid_o 1
  }


}
proc lm:vo:set_message text {

  global lm_vo_msg

  set lm_vo_msg $text


}
proc lm:vo:disable {} {

  foreach ch [fd:all_channels_for_floortype V] {
    set f [fd:id_from_channeltype $ch V]
    fu:set_mute W $f
  }
}
proc lm:vo:set_vsizebar s {

  global lm_vo_size

  set lm_vo_size $s

}

set Name $Parent.f_vid_out.l163
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .media1.f_vid_out.l163 <ButtonPress-
1>}}

set Name $Parent.f_vid_out.s164
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.f_vid_out.l165
# Procedures----------------------------

set Name $Parent.f_vid_out.b166
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.f_vid_out.b167
```

```
# Procedures----------------------------

set Name $Parent.f_vid_out.lmsg
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.f_audio_in2
# Procedures----------------------------
global Procs
set Procs($Name) { lm:ai:enable lm:ai:toggle_frame
lm:ai:set_message lm:ai:set_source lm:ai:show_source
lm:ai:show_volume}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) lm:ai:
set Glob_prefix($Name) lm_ai_
proc lm:ai:enable {} {

 global lm_ai_bmute

    if {$lm_ai_bmute} {
      foreach ch [fd:all_channels_for_floortype A] {
        set f [fd:id_from_channeltype $ch A]
        fu:set_mute R $f
      }
    } else {
    set succes 0
    foreach ch [fd:all_channels_for_floortype A] {
      set f [fd:id_from_channeltype $ch A]
      set succes [expr [fu:clear_mute R $f] || $succes]
    }
    set lm_ai_bmute  [expr !$succes]
  }
}
proc lm:ai:toggle_frame {} {

  global lm_shrunk_au_i lm_frame_au_i

  if {$lm_shrunk_au_i} {
    lm:frame_expand $lm_frame_au_i
    set lm_shrunk_au_i 0
  } else {
    lm:frame_shrink $lm_frame_au_i
    set lm_shrunk_au_i 1
  }
}
proc lm:ai:set_message text {

  global lm_ai_msg

  set lm_ai_msg $text

}
proc lm:ai:set_source {} {

  #set destination according to the checkboxes

  global lm_ai_speaker lm_ai_head lm_ai_aux

  sy:set_audio_out_source $lm_ai_speaker $lm_ai_head $lm_ai_aux

}
proc lm:ai:show_source {speaker phone aux} {

  #set values of audio-destination checkboxes

  global lm_ai_speaker lm_ai_head lm_ai_aux

  set lm_ai_speaker $speaker
  set lm_ai_head $phone
  set lm_ai_aux $aux
  update idletasks
}
proc lm:ai:show_volume vol {

  global lm_ai_volume

  set lm_ai_volume $vol
}

set Name $Parent.f_audio_in2.l172
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .media1.f_audio_in2.l172 <ButtonPress-
1>}}

set Name $Parent.f_audio_in2.s173
# Procedures----------------------------

set Name $Parent.f_audio_in2.l174
# Procedures----------------------------

set Name $Parent.f_audio_in2.b178
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.f_audio_in2.b137
# Procedures----------------------------

set Name $Parent.f_audio_in2.b139
# Procedures----------------------------

set Name $Parent.f_audio_in2.b141
# Procedures----------------------------

set Name $Parent.f_audio_in2.l_msg
# Procedures----------------------------

set Name $Parent.f_audio_out
# Procedures----------------------------
global Procs
set Procs($Name) { lm:ao:enable lm:ao:toggle_frame
lm:ao:set_message lm:ao:use_mute lm:ao:use_speak
lm:ao:set_source lm:ao:show_source lm:ao:show_volumes}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) lm:ao:
set Glob_prefix($Name) lm_ao_
proc lm:ao:enable {} {

  global lm_ao_bmute

    if {$lm_ao_bmute} {
      foreach ch [fd:all_channels_for_floortype A] {
        set f [fd:id_from_channeltype $ch A]
        fu:set_mute W $f
      }
    } else {
      set succes 0
      foreach ch [fd:all_channels_for_floortype A] {
        set f [fd:id_from_channeltype $ch A]
        set succes [expr [fu:clear_mute W $f] || $succes]
      }
      set lm_ao_bmute [expr !$succes]
   }
}
proc lm:ao:toggle_frame {} {

  global lm_shrunk_au_o lm_frame_au_o

  if {$lm_shrunk_au_o} {
    lm:frame_expand $lm_frame_au_o
    set lm_shrunk_au_o 0
  } else {
    lm:frame_shrink $lm_frame_au_o
    set lm_shrunk_au_o 1
  }

}
proc lm:ao:set_message text {

  global lm_ao_msg

  set lm_ao_msg $text

}
proc lm:ao:use_mute {} {

  whide .media1.f_audio_out.b_speak
  wshow .media1.f_audio_out.b_mute

}
proc lm:ao:use_speak {} {

  global lm_ao_bmute

  whide .media1.f_audio_out.b_mute
  wshow .media1.f_audio_out.b_speak

  set lm_ao_bmute 1

}
proc lm:ao:set_source {} {

  global lm_ao_source

  sy:set_audio_in_source $lm_ao_source

}
proc lm:ao:show_source source {

  #show selected source in radiobuttons

  global lm_ao_source

  set lm_ao_source $source
}
proc lm:ao:show_volumes {vol_s vol_m} {

global lm_ao_volume_src lm_ao_volume_mon

  set lm_ao_volume_src $vol_s
  set lm_ao_volume_mon $vol_m

}

set Name $Parent.f_audio_out.b181
# Procedures----------------------------

set Name $Parent.f_audio_out.b182
# Procedures----------------------------

set Name $Parent.f_audio_out.l184
# Procedures----------------------------

set Name $Parent.f_audio_out.s185
# Procedures----------------------------

set Name $Parent.f_audio_out.l188
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .media1.f_audio_out.l188 <ButtonPress-
1>}}

set Name $Parent.f_audio_out.s206
# Procedures----------------------------

set Name $Parent.f_audio_out.l207
# Procedures----------------------------

set Name $Parent.f_audio_out.l_msg
# Procedures----------------------------

set Name $Parent.f_audio_out.b_speak
```

```
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .media1.f_audio_out.b_speak
<ButtonPress-1>} {bind .media1.f_audio_out.b_speak
<ButtonRelease-1>}}

set Name $Parent.f_audio_out.b_mute
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.f_con
# Procedures----------------------------
global Procs
set Procs($Name) { lm:frame_shrink lm:frame_expand lm:init_win
lm:exit_win lm:frame_hide lm:hide_floors lm:show_floors
lm:show_win media_show lm:hide_win lm:show_muting
lm:recover_all}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) lm:
set Glob_prefix($Name) lm_
proc lm:frame_shrink frame {

  global lm_new_h

  set lm_new_h($frame) 25
  media_show $frame
}
proc lm:frame_expand frame {

  global lm_old_h lm_new_h lm_mshow

  set lm_new_h($frame) $lm_old_h($frame)
  media_show $frame

 }
proc lm:init_win {half_dupl s p} {

  global lm_frame_vid_i lm_frame_vid_o lm_frame_au_i
lm_frame_au_o
  global lm_shrunk_vid_i lm_shrunk_vid_o lm_shrunk_au_i
lm_shrunk_au_o
  global lm_old_h lm_new_h lm_mshow lm_mstart
  global lm_vo_size lm_vo_preview
  global lm_vo_change

  set lm_frame_vid_i .media1.f_vid_in
  set lm_frame_vid_o .media1.f_vid_out
  set lm_frame_au_i .media1.f_audio_in2
  set lm_frame_au_o .media1.f_audio_out

  set lm_shrunk_vid_i 0
  set lm_shrunk_vid_o 0
  set lm_shrunk_au_i 0
  set lm_shrunk_au_o 0

  set lm_vo_change 0

  set lm_mstart 1

  #media-window
  wincreate .media1
  wshow .media1
  update

  set lm_old_h($lm_frame_vid_i) [winfo heigh $lm_frame_vid_i]
  set lm_old_h($lm_frame_vid_o) [winfo heigh $lm_frame_vid_o]
  set lm_old_h($lm_frame_au_i) [winfo heigh $lm_frame_au_i]
  set lm_old_h($lm_frame_au_o) [winfo heigh $lm_frame_au_o]

  if {$half_dupl} {
    lm:ao:use_speak
  } else {
    lm:ao:use_mute
  }

  set lm_mshow 0

  lm:frame_hide $lm_frame_vid_i
  lm:frame_hide $lm_frame_vid_o
  lm:frame_hide $lm_frame_au_i
  lm:frame_hide $lm_frame_au_o

  whide .media1

  #hide all frames
  #lm:hide_floors A
  #lm:hide_floors V

  #set video-out size and preview
  set lm_vo_size $s
  set lm_vo_preview $p
  lm:vo:set_vsizebar [nl:prepare_vid_out_props $lm_vo_size
$lm_vo_preview]


}
proc lm:exit_win {} {

  windestroy .media1

}
proc lm:frame_hide frame {

  global lm_new_h

  set lm_new_h($frame) 0

  media_show $frame
}
proc lm:hide_floors {dir m} {

  global lm_frame_vid_i lm_frame_vid_o lm_frame_au_i
lm_frame_au_o


  if {$m == "A"} {
    if {$dir != "W"} {
      lm:frame_hide $lm_frame_au_i
    }
    if {$dir != "R"} {
      lm:frame_hide $lm_frame_au_o
    }
  }
  if {$m == "V"} {
    if {$dir != "W"} {
      lm:frame_hide $lm_frame_vid_i
    }
    if {$dir != "R"} {
      lm:frame_hide $lm_frame_vid_o
    }
  }

}
proc lm:show_floors {dir m} {

  global lm_frame_vid_i lm_frame_vid_o lm_frame_au_i
lm_frame_au_o


  if {$m == "A"} {
    if {$dir != "W"} {
      lm:frame_expand $lm_frame_au_i
    }
    if {$dir != "R"} {
      lm:frame_expand $lm_frame_au_o
    }
  }
  if {$m == "V"} {
    if {$dir != "W"} {
      lm:frame_expand $lm_frame_vid_i
    }
    if {$dir != "R"} {
      lm:frame_expand $lm_frame_vid_o
    }
  }
}
proc lm:show_win {} {

  global lm_frame_vid_i lm_frame_vid_o lm_frame_au_i
lm_frame_au_o
  global lm_mshow lm_mstart

  wshow .media1

  set lm_mshow 1
  set lm_mstart 0

  media_show $lm_frame_vid_i
  media_show $lm_frame_vid_o
  media_show $lm_frame_au_i
  media_show $lm_frame_au_o

}
proc media_show frame {

  global lm_new_h lm_mshow

   if {$lm_mshow} {
    $frame configure -height $lm_new_h($frame)
  }
  focus .media1
}
proc lm:hide_win {} {

  global lm_mshow

  set lm_mshow 0
  whide .media1

}
proc lm:show_muting {} {

  global vid_in_button_mute
  global vid_out_button_mute
  global audio_in_button_mute
  global audio_out_button_mute

  #Must be changed when there is more than one audio-channel

  foreach ch [fd:all_channels_for_floortype V] {
    set f [fd:id_from_channeltype $ch V]
    set vid_in_button_mute [fd:is_muted R $f]
    set vid_out_button_mute [fd:is_muted W $f]
  }

  foreach ch [fd:all_channels_for_floortype A] {
    set f [fd:id_from_channeltype $ch A]
    set audio_in_button_mute [fd:is_muted R $f]
    set audio_out_button_mute [fd:is_muted W $f]
  }
}
proc lm:recover_all {} {

  #restore audio
  lm:ao:enable

  #restore video
  lm:vo:enable
  lm:vi:enable
}

set Name $Parent.f_con.b190
# Procedures----------------------------
```

```
global Procs
set Procs($Name) {{bind .media1.f_con.b190 <KeyPress-Return>}}

set Name $Parent.dummy
# Procedures----------------------------
#- TOP LEVEL procedures---------------------------------

set Name .supercontrol
set Parent $Name
global Procs
set Procs($Name) { w_sc:init_win w_sc:exit_win
w_sc:pipe_command w_sc:tcl_command w_sc:addlst_pipe_out
w_sc:addlst_pipe_in w_sc:addlst_debug w_sc:clearlists}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) w_sc:
set Glob_prefix($Name) w_sc_
proc w_sc:init_win debug {

  global sc_lb_pipe_out sc_lb_pipe_in sc_lb_debug
  global sc_debug

  set sc_debug $debug

  set sc_lb_pipe_out ".supercontrol.f_out.lb"
  set sc_lb_pipe_in ".supercontrol.f_in.lb"
  set sc_lb_debug ".supercontrol.f_debug.lb"

  #control optional
  if {$sc_debug} {
    wincreate .supercontrol
    winicon .supercontrol
  }
}
proc w_sc:exit_win {} {

  global sc_debug

  if {$sc_debug} {
    windestroy .supercontrol
  }
}
proc w_sc:pipe_command {} {

global sc_pcommand

main:status_message "Executing command"
update

sy:pipe_write $sc_pcommand

set sc_pcommand ""

main:status_message "READY"
update
}
proc w_sc:tcl_command {} {

global sc_tcommand

main:status_message "Executing Tcl/TK command"
update

#execute command in global environment
uplevel #0 {eval $sc_tcommand}

set sc_tcommand ""

main:status_message "READY"
update



}
proc w_sc:addlst_pipe_out text {

  global sc_lb_pipe_out sc_debug

  if {!$sc_debug} {return}

  catch {$sc_lb_pipe_out insert end "$text"}

  #scroll to new text
  catch {$sc_lb_pipe_out yview end}

  update

}
proc w_sc:addlst_pipe_in text {

  global sc_lb_pipe_in sc_debug

  if {!$sc_debug} {return}

  $sc_lb_pipe_in insert end "$text"

  #scroll to new text
  $sc_lb_pipe_in yview end
  update

}
proc w_sc:addlst_debug text {

  global sc_lb_debug sc_debug

  if {!$sc_debug} {return}

  catch {$sc_lb_debug insert end "$text"}

  #scroll to new text
  catch {$sc_lb_debug yview end}
  update
}
proc w_sc:clearlists {} {

  global sc_lb_pipe_out sc_lb_pipe_in sc_lb_debug
  global sc_debug

  if {!$sc_debug} {return}

  $sc_lb_pipe_out delete 0 end
  $sc_lb_pipe_in delete 0 end
  $sc_lb_debug delete 0 end

}

set Name $Parent.l30
# Procedures----------------------------

set Name $Parent.e27
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .supercontrol.e27 <KeyPress-Return>}}

set Name $Parent.l29
# Procedures----------------------------

set Name $Parent.b45
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.f_out
# Procedures----------------------------

set Name $Parent.f_out.l139
# Procedures----------------------------

set Name $Parent.f_out.sb140
# Procedures----------------------------

set Name $Parent.f_out.lb
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.f_in
# Procedures----------------------------

set Name $Parent.f_in.l144
# Procedures----------------------------

set Name $Parent.f_in.sb146
# Procedures----------------------------

set Name $Parent.f_in.lb
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.f_debug
# Procedures----------------------------

set Name $Parent.f_debug.l128
# Procedures----------------------------

set Name $Parent.f_debug.sb129
# Procedures----------------------------

set Name $Parent.f_debug.lb
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.e206
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .supercontrol.e206 <KeyRelease-
Return>}}

set Name $Parent.l208
# Procedures----------------------------
#- TOP LEVEL procedures---------------------------------

set Name .text1
set Parent $Name
global Procs
set Procs($Name) { w_txt:displaytext w_txt:display_incoming
w_txt:send_entry w_txt:winname w_txt:sendtext w_txt:clear_entry
w_txt:clear_textdisplay w_txt:create w_txt:hide w_txt:destroy
w_txt:in_message w_txt:out_message w_txt:show w_txt:show_all }
global Proc_prefix Glob_prefix
set Proc_prefix($Name) w_txt:
set Glob_prefix($Name) w_txt_
proc w_txt:displaytext {channel from text color} {

  global w_txt_fb_out

  #show text on window

  #determine wich widget to use
  set lb_textout "[w_txt:winname $channel]$w_txt_fb_out"

  #set text
  $lb_textout insert end "$from: $text"
  #change color
  $lb_textout item configure end -foreground $color

  #scroll to new text
  $lb_textout yview end

}
proc w_txt:display_incoming {channel ipnr text} {
```

```
  w_txt:displaytext $channel "[ud:name_in_meetinglist $ipnr 1]"
"$text" black
}
proc w_txt:send_entry channel {

  global w_txt_entry

  #get widget
  set text_entry [w_txt:winname $channel]$w_txt_entry

  #get text
  set text "[$text_entry get]"
  #send text
  w_txt:sendtext $channel $text

  #clear text
  w_txt:clear_entry $channel

}
proc w_txt:winname channel {

  global w_txt_winbase

  return [win_name $w_txt_winbase $channel]
}
proc w_txt:sendtext {channel text} {

  global w_txt_fb_out

  #if transmit and display text, if transmission possible
  if {[fu:text_out $channel $text]} {
    w_txt:displaytext $channel *[ud:get_current_username 1]
"$text" blue
  }
}
proc w_txt:clear_entry channel {

  global w_txt_entry

  #clear text on window

  #get widget
  set text_entry [w_txt:winname $channel]$w_txt_entry

  $text_entry delete 0 end


}
proc w_txt:clear_textdisplay nr {

  global w_txt_fb_out

  #clear text on window

  #determine wich widget to use
  set lb_textout "[w_txt:winname $nr]$w_txt_fb_out"

  $lb_textout delete 0 end
}
proc w_txt:create {channel name} {

  global w_txt_winbase w_txt_entry w_txt_fb_out w_txt_in_msg
w_txt_out_msg

  set w_txt_winbase ".text1"
  set w_txt_entry ".entry"
  set w_txt_fb_out ".fb_textout"
  set w_txt_in_msg ".in_message"
  set w_txt_out_msg ".l_out_msg"

  wincreate "$w_txt_winbase" "$channel"
  w_txt:hide $channel

  #determine wich widget to use
  set win "[w_txt:winname $channel]"

  $win$w_txt_entry configure -textvariable text$channel

  #index for window cannot directly in main-procedure
  bind $win$w_txt_entry <KeyRelease-Return> "w_txt:send_entry
$channel"
  set cmd "$win.b_send configure -command \"w_txt:send_entry
$channel\""
  eval $cmd
  set cmd "bind $win.b_send <KeyPress-Return> \{%W invoke\}"
  eval $cmd
  bind $win <Alt-s> "w_txt:send_entry $channel"

  set cmd "$win.b_hide configure -command \"w_txt:hide
$channel\""
  eval $cmd
  set cmd "bind $win.b_c_text <KeyPress-Return> \{%W invoke\}"
  eval $cmd
  bind $win <Alt-h> "w_txt:hide $channel"

  set cmd "$win.b_c_entry configure -command
\"w_txt:clear_entry $channel\""
  eval $cmd
  set cmd "bind $win.b_c_entry <KeyPress-Return> \{%W invoke\}"
  eval $cmd
  bind $win <Alt-e> "w_txt:clear_entry $channel"

  set cmd "$win.b_c_text configure -command
\"w_txt:clear_textdisplay $channel\""
  eval $cmd
  set cmd "bind $win.b_c_text <KeyPress-Return> \{%W invoke\}"
  eval $cmd
  bind $win <Alt-c> "w_txt:clear_textdisplay $channel"


  #set title
  wm title $win "$name"
}
proc w_txt:hide channel {

  #determine wich widget to use
  set win "[w_txt:winname $channel]"

  whide $win

}
proc w_txt:destroy channel {

  #determine wich widget to use
  set win "[w_txt:winname $channel]"

  windestroy $win

}
proc w_txt:in_message {channel text} {

  global w_txt_in_msg

  #get widget
  set text_msg [w_txt:winname $channel]$w_txt_in_msg

  if {$text != ""} {
    $text_msg configure -text "$text"
  }

}
proc w_txt:out_message {channel text} {

  global w_txt_out_msg

  #get widget
  set text_msg [w_txt:winname $channel]$w_txt_out_msg

  if {$text != ""} {
    $text_msg configure -text "$text"
  }

}
proc w_txt:show channel {

  #determine wich widget to use
  set win "[w_txt:winname $channel]"
  wshow $win
  focus $win.entry

}
proc w_txt:show_all {} {

  set textc [fd:all_channels_for_floortype T]
  foreach t $textc {
    w_txt:show $t
  }
}

set Name $Parent.sb96
# Procedures----------------------------

set Name $Parent.l_out_msg
# Procedures----------------------------

set Name $Parent.entry
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.in_message
# Procedures----------------------------

set Name $Parent.fb_textout
# Procedures----------------------------

set Name $Parent.b_hide
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.b_c_entry
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.b_send
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.b_c_text
# Procedures----------------------------
global Procs
set Procs($Name) { }
#- TOP LEVEL procedures---------------------------------

set Name .freq_popup
set Parent $Name
global Procs
set Procs($Name) { w_freq_hide w_freq_apply w_freq_show
w_freq_getflist w_freq_showfmenu w_freq_floor_usable {bind
.freq_popup <Alt-o>} {bind .freq_popup <Alt-c>} {bind
.freq_popup <Alt-f>} {bind .freq_popup <Alt-r>} {bind
.freq_popup <KeyPress-Escape>}}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) w_freq:
set Glob_prefix($Name) w_freq_
proc w_freq_hide {} {

global w_freq_reason

  catch {
```

```
   grab release .freq_popup
   windestroy .freq_popup
  }
}
proc w_freq_apply {} {

  global w_freq_real_choice w_freq_reason

  w_fc:request_floor $w_freq_real_choice "$w_freq_reason"
  w_freq_hide

}
proc w_freq_show {} {

  global w_freq_menu_parent w_freq_real_choice w_freq_menu

  set w_freq_menu_parent ".freq_popup.m_floormenu"
  set w_freq_menu ".freq_popup.chose.m"

  #Empty reason
  set w_freq_reason ""
  wincreate .freq_popup
  focus .freq_popup.entry1
  grab .freq_popup

  w_freq_getflist

}
proc w_freq_getflist {} {

  #get all 1 and 2 combinations of floors
  #and an entry for all floors (R+W, R)
  #and put these in a menu

  global w_freq_menu_parent w_freq_real_choice
w_freq_choiceview

  set ctrl [fd:get_fcontrollers [fd:all_floor_ids]]

  set total ""
  foreach c $ctrl {

    #first get all floors, controlled by controller c
    set cf [fd:get_floors_controlled_by $c]
    set fl_e ""

    #write+read becomes read & write+read
    foreach f $cf {
      if {[fd:dir $f] == "X"} {
        lappend fl_e "R[fd:id $f]"
      }
    }

    #if there are more then 2 floors,
    #start with an entry for all floors (read + write)
    if {$cf != "" && [llength $cf] > 2} {
      lappend total $cf
    }

    #if there are more than 2 extra floor-entrys,
    #make with an entry with only read
    #when there are more than two extra entry's
    if {$fl_e != "" && [llength $fl_e] > 2} {
       lappend total $fl_e
    }

    set af [w_freq_floor_usable [concat $cf $fl_e]]

    #make combinations of 2 floors and add these.
    #make sure that there are no combinations with
    #the same floor-ids
    for {set t 0} {$t < [llength $af]} {incr t 1} {
      if {[llength $af] > 0} {
        if {[llength $af] > 1} {
          for {set i [expr $t +1]} {$i < [llength $af]} {incr i
1} {
            set f1 [lindex $af $t]
            set f2 [lindex $af $i]
            if {[fd:id $f1] != [fd:id $f2]} {
              lappend total "$f1 $f2"
            }
          }
        }
        lappend total [lindex $af $t]
      }
    }
  }

  #set initial value for menu-choice
  set w_freq_real_choice [lindex $total 0]
  set w_freq_choiceview [fh:get_full_floormanes
$w_freq_real_choice]

  #make into menu-entrys
  set t_menu ""
  foreach fn $total {
    set names [fh:get_full_floormanes $fn]
    lappend t_menu "\{$names\} command \{set w_freq_choiceview
\{$names\}; set w_freq_real_choice \{$fn\}\}"
  }

  set parent $w_freq_menu_parent

  #this procedure is to slow to invoke when the menu is
  #called, so it must be predefined.

  #create menu definition
  set Menu_string($parent) "\{$t_menu\} -tearoff 0"

 #make menu itself

 eval "make_menu $parent $Menu_string($parent)"
}
proc w_freq_showfmenu {} {

  global w_freq_menu_parent w_freq_menu
  set ystart [winfo rooty "$w_freq_menu_parent"]
  set xstart [winfo rootx "$w_freq_menu_parent"]
  set x $xstart
  set y $ystart
  tk_popup $w_freq_menu $x $y

}
proc w_freq_floor_usable fn {

    set fu ""
    foreach f $fn {
      set dir [fd:dir $f]
      set fo [fd:id $f]
      if {$dir == "X"} {
        if {[fd:is_muted W $fo] >= 0} {
          if {[fd:is_muted R $fo] >= 0} {
            lappend fu $f
          }
        }
      } else {
        if {[fd:is_muted $dir $fo] >= 0} {
          lappend fu $f
        }
      }
    }
  return $fu
}

set Name $Parent.b91
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .freq_popup.b91 <KeyPress-Return>}}

set Name $Parent.l103
# Procedures----------------------------

set Name $Parent.l106
# Procedures----------------------------

set Name $Parent.b110
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .freq_popup.b110 <KeyPress-Return>}}

set Name $Parent.entry1
# Procedures----------------------------
global Procs
set Procs($Name) { {bind .freq_popup.entry1 <KeyPress-Return>}}

set Name $Parent.m_floormenu
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .freq_popup.m_floormenu <KeyPress-
Return>}}
#- TOP LEVEL procedures---------------------------------

set Name .fref_popup
set Parent $Name
global Procs
set Procs($Name) {w_fp:ok w_fp:hide w_fp:show {bind .fref_popup
<Alt-c>} {bind .fref_popup <Alt-o>} {bind .fref_popup
<KeyPress-Escape>} w_fp:init}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) w_fp:
set Glob_prefix($Name) w_fp_
proc w_fp:ok {} {

  global w_fp_reason w_fp_media w_fp_ipnr

  w_fc:refuse_floor $w_fp_media $w_fp_ipnr "$w_fp_reason"
  w_fp:hide
}
proc w_fp:hide {} {

  grab release .fref_popup
  windestroy .fref_popup
}
proc w_fp:show {} {

  wincreate .fref_popup

  focus .fref_popup.entry1
  grab .fref_popup
}
proc w_fp:init {media ipnr} {

  global w_fp_reason w_fp_media w_fp_ipnr

  set w_fp_reason ""
  set w_fp_media $media
  set w_fp_ipnr $ipnr
}

set Name $Parent.l102
# Procedures----------------------------

set Name $Parent.b103
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .fref_popup.b103 <KeyPress-Return>}}

set Name $Parent.b104
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .fref_popup.b104 <KeyPress-Return>}}

set Name $Parent.entry1
```

```
# Procedures----------------------------
global Procs
set Procs($Name) { {bind .fref_popup.entry1 <KeyPress-Return>}}
#- TOP LEVEL procedures---------------------------------

set Name .groupmgr
set Parent $Name
global Procs
set Procs($Name) {w_gmgr:init {bind .groupmgr <Alt-h>}
w_gmgr:show w_gmgr:hide}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) w_grmgr:
set Glob_prefix($Name) w_grmgr_
proc w_gmgr:init {} {

  #Groupmeeting manager
  wincreate .groupmgr
  whide .groupmgr
}
proc w_gmgr:show {} {

  wshow .groupmgr
  focus .groupmgr
  focus .groupmgr.b_hide

}
proc w_gmgr:hide {} {

  whide .groupmgr

}

set Name $Parent.f128
# Procedures----------------------------

set Name $Parent.f128.fb134
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.f128.l135
# Procedures----------------------------

set Name $Parent.f128.sb137
# Procedures----------------------------

set Name $Parent.f138
# Procedures----------------------------

set Name $Parent.f138.fb180
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.f138.l181
# Procedures----------------------------

set Name $Parent.f138.sb183
# Procedures----------------------------

set Name $Parent.l155
# Procedures----------------------------

set Name $Parent.l157
# Procedures----------------------------

set Name $Parent.l160
# Procedures----------------------------

set Name $Parent.l162
# Procedures----------------------------

set Name $Parent.f157
# Procedures----------------------------
global Procs
set Procs($Name) { }
global Proc_prefix Glob_prefix
set Proc_prefix($Name) w_grmgr:f157:
set Glob_prefix($Name) w_grmgr_f157_

set Name $Parent.f157.l158
# Procedures----------------------------

set Name $Parent.f157.m159
# Procedures----------------------------
global Procs
set Procs($Name) {w_gmgr:req_floor_audio
w_gmgr:req_floor_au_txt w_gmgr:req_floor_text
w_gmgr:lower_freq_button w_gmgr:raise_freq_button}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) w_grmgr:f157:m159:
set Glob_prefix($Name) w_grmgr_f157_m159_
proc w_gmgr:req_floor_audio {} {

request_floor A1 "xx"

}
proc w_gmgr:req_floor_au_txt {} {

  request_floor A1+T1 xx

}
proc w_gmgr:req_floor_text {} {

request_floor T1 xx

}
proc w_gmgr:lower_freq_button {} {

  global b_floor_req b_floor_withdraw

  $b_floor_req configure -relief sunken
  $b_floor_req configure -state disabled
  $b_floor_withdraw configure -state normal

}
proc w_gmgr:raise_freq_button {} {

  global b_floor_req b_floor_withdraw

  $b_floor_req configure -relief raised
  $b_floor_withdraw configure -state disabled

}

set Name $Parent.f157.b160
# Procedures----------------------------

set Name $Parent.f157.b161
# Procedures----------------------------

set Name $Parent.f157.b162
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.f157.l163
# Procedures----------------------------

set Name $Parent.b165
# Procedures----------------------------

set Name $Parent.b_hide
# Procedures----------------------------
#- TOP LEVEL procedures---------------------------------

set Name .trans_start
set Parent $Name
global Procs
set Procs($Name) { w_st1:init w_st1:start_conference w_st1:show
w_st1:quit w_st1:stm_show w_st1:atm_set w_st1:atm_unset
w_st1:hide {bind .trans_start <Alt-s>} {bind .trans_start <Alt-
q>} {bind .trans_start <Key-Return>}}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) w_st1:
set Glob_prefix($Name) w_st1_
proc w_st1:init {m_nr u_nr no_c u_atm dbg vc vs vp sp sl_emu} {

  global wst1_is_teacher wst1_id_input wst1_refl_address
  global wst1_atm_entry wst1_use_atm wst1_atm_entryvalue
  global wst1_coding
  global wst1_winname wst1_enable_debug
  global wst1_no_camera wst1_half_dupl
  global wst1_vo_size wst1_preview
  global wst1_pict
  global  wst1_sl_emu

  #m_nr = meeting number
  #u_nr = user number
  #no_c = No Camera
  #u-atm = use IP over ATM
  #vc = Video-Codec
  #vs = Video Size
  #vp = Video Preview
  #sp = Show Pictures

  set wst1_winname .trans_start
  wincreate $wst1_winname


  set wst1_refl_address "newton"
  set wst1_id_input "student"
  set wst1_is_teacher 0

  if {$u_nr != ""} {
    set wst1_id_input $u_nr
  }

  if {$m_nr != ""} {
    set wst1_refl_address $m_nr
  }

  set wst1_atm_entry ".trans_start.e_atm"
  whide $wst1_atm_entry

  set wst1_use_atm $u_atm
  set wst1_atm_entryvalue ""

  set wst1_enable_debug $dbg
  set wst1_no_camera $no_c
  set wst1_half_dupl 0

  #initialize parameters
  set wst1_coding $vc
  set wst1_vo_size $vs
  set wst1_preview $vp
  set wst1_pict $sp
  set wst1_sl_emu $sl_emu

  main:status_message "READY"
}
proc w_st1:start_conference {} {

  global wst1_coding wst1_id_input wst1_refl_address
wst1_startpressed wst1_enable_debug
  global no_mm wst1_is_teacher vid_uncontrol wst1_use_atm
wst1_atm_entryvalue
  global wst1_no_camera wst1_half_dupl
  global wst1_winname
  global wst1_vo_size wst1_preview
  global wst1_pict
  global  wst1_sl_emu

  catch {if {$wst1_startpressed} {return}}
```

```
  set wst1_startpressed 1

  main:status_message "Starting conference"

   if {$wst1_use_atm} {
     set a_adr [string trim $wst1_atm_entryvalue]
   } else {
      set a_adr ""
   }

  if {[sy:check_address $wst1_refl_address $a_adr
$wst1_sl_emu]} {
   #all is well !
   #initialize everything and start network layer
   w_sc:init_win $wst1_enable_debug
   set audio_fail [sy:init_audio]
   nl:init_vid_coding $wst1_coding
   nl:set_vidr_auto 1

   ud:set_current_username $wst1_id_input -1

   lm:init_win $wst1_half_dupl $wst1_vo_size $wst1_preview

   nl:start_network $wst1_id_input $wst1_refl_address $a_adr
$wst1_sl_emu

   ud:use_no_userdatabase

   w_fc:first_initialisation "w_st1:show" $wst1_enable_debug

   fr:set_floor_db $no_mm $vid_uncontrol $wst1_is_teacher
$wst1_refl_address

   w_fc:start $wst1_pict

   #hide startwindow
   w_st1:hide
   set wst1_startpressed 0

   #check for failures and enable floor-reception

   if {$audio_fail} {
     fu:audio_unusable
   }
   if {$wst1_no_camera} {
     fu:video_write_unusable
   }
   foreach f [fd:all_floor_ids] {
     fu:clear_mute W $f
     fu:clear_mute R $f
   }
   lm:show_muting

   nl:enable_reception
  }
}
proc w_st1:show {} {

  global wst1_use_atm wst1_atm_entry

  wincreate .trans_start
  wshow .trans_start

  if {!$wst1_use_atm} {
    whide $wst1_atm_entry
  }

  focus .trans_start
  focus .trans_start.e_n
}
proc w_st1:quit {} {

main:status_message "Quiting"
sy:quit_all

}
proc w_st1:stm_show {} {

global wst1_use_atm wst1_atm_entry

if {$wst1_use_atm} then {
  wshow $wst1_atm_entry
  w_st1:atm_set
} else {
  whide $wst1_atm_entry
  w_st1:atm_unset
}
}
proc w_st1:atm_set {} {

  global atm_adr wst1_refl_address

  set wst1_refl_address [string trim $wst1_refl_address]

  set atm_adr [sy:get_atm_adr [sy:get_local_adr]]
  set wst1_refl_address [sy:get_atm_adr $wst1_refl_address]

}
proc w_st1:atm_unset {} {

  global wst1_refl_address

  set wst1_refl_address [sy:get_eth_adr [string trim
$wst1_refl_address]]
}
proc w_st1:hide {} {

  windestroy .trans_start

}

set Name $Parent.l_message
# Procedures----------------------------

set Name $Parent.l6
# Procedures----------------------------

set Name $Parent.e_refaddr
# Procedures----------------------------
global Procs
set Procs($Name) { {bind .trans_start.e_refaddr <KeyPress>}}

set Name $Parent.l11
# Procedures----------------------------

set Name $Parent.b_go
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.b_quit
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.l13
# Procedures----------------------------

set Name $Parent.f0
# Procedures----------------------------

set Name $Parent.f0.lb1
# Procedures----------------------------

set Name $Parent.f0.sb2
# Procedures----------------------------

set Name $Parent.b_teacher
# Procedures----------------------------
global Procs
set Procs($Name) { {bind .trans_start.b_teacher <ButtonRelease-
1>}}

set Name $Parent.f153
# Procedures----------------------------

set Name $Parent.f153.b155
# Procedures----------------------------

set Name $Parent.f153.b157
# Procedures----------------------------

set Name $Parent.f153.b159
# Procedures----------------------------

set Name $Parent.f153.l161
# Procedures----------------------------

set Name $Parent.f153.b153
# Procedures----------------------------

set Name $Parent.f153.b154
# Procedures----------------------------

set Name $Parent.b121
# Procedures----------------------------

set Name $Parent.b126
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.e_atm
# Procedures----------------------------

set Name $Parent.b128
# Procedures----------------------------

set Name $Parent.b185
# Procedures----------------------------

set Name $Parent.b193
# Procedures----------------------------

set Name $Parent.b197
# Procedures----------------------------

set Name $Parent.e_n
# Procedures----------------------------

set Name $Parent.b201
# Procedures----------------------------
#- TOP LEVEL procedures---------------------------------

set Name .floor_info
set Parent $Name
global Procs
set Procs($Name) { w_fi:show w_fi:hide w_fi:init w_fi:destroy
{bind .floor_info <Alt-h>} {bind .floor_info <KeyPress-Return>}
{bind .floor_info <KeyPress-Escape>} w_fi:show_floor_info}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) w_fi:
set Glob_prefix($Name) w_fi_
proc w_fi:show {} {

  wshow .floor_info
  focus .floor_info
}
proc w_fi:hide {} {

  whide .floor_info

}
proc w_fi:init {} {
```

```
  global w_fi_w_fi_flistbox w_fi_lbframe w_fi_flist

  wincreate .floor_info
  w_fi:hide

  set w_fi_w_fi_flistbox ".floor_info.f.fb1"
  set w_fi_lbframe ".floor_info.f"
  set w_fi_flist ""

  w_fi:list:clear

}
proc w_fi:destroy {} {

  catch {windestroy .floor_info}

}
proc w_fi:show_floor_info media {

  foreach m $media {
    set f [fd:id $m]
    #get tokens
    set f_rd [fd:has_token R $f]
    set f_wr [fd:has_token W $f]
    #is floor usable ?
    if {[fd:is_muted R $f] == -1} {
       set f_rd -1
    }
    if {[fd:is_muted W $f] == -1} {
       set f_wr -1
    }
    #has floor been requested
    set f_rd_req [expr [lsearch -exact
[rg:get_requested_floors] "R$f"] >= 0]
    set f_wr_req [expr [lsearch -exact
[rg:get_requested_floors] "W$f"] >= 0]
    w_fi:list:set_props $f [fd:name $f]
[fd:check_user_is_fc_for_floors 0 $f] $f_rd $f_wr 0 $f_rd_req
$f_wr_req
  }
}

set Name $Parent.f
# Procedures----------------------------

set Name $Parent.f.sb178
# Procedures----------------------------

set Name $Parent.f.fb1
# Procedures----------------------------
global Procs
set Procs($Name) { w_fi:list:add w_fi:list:del w_fi:list:clear
w_fi:list:set_props }
global Proc_prefix Glob_prefix
set Proc_prefix($Name) w_fi:list:
set Glob_prefix($Name) w_fi_list_
proc w_fi:list:add f {

  global w_fi_w_fi_flistbox w_fi_lbframe w_fi_flist

  lappend w_fi_flist $f

  #empty line: use w_fi_flist_props to fill this in.
  $w_fi_w_fi_flistbox insert end " "

}
proc w_fi:list:del f {

  global w_fi_flist w_fi_w_fi_flistbox

  set lnr [lsearch -glob $w_fi_flist $f]
  if {$lnr >= 0} {
    $w_fi_w_fi_flistbox delete $lnr
    destroy $w_fi_w_fi_flistbox.lr$lnr
    destroy $w_fi_w_fi_flistbox.lw$lnr
    destroy $w_fi_w_fi_flistbox.lc$lnr

  }


}
proc w_fi:list:clear {} {

  global w_fi_flist w_fi_w_fi_flistbox

  catch {$w_fi_w_fi_flistbox delete 0 end}
  set w_fi_flist ""


}
proc w_fi:list:set_props {f name ctrl rd wr ctrl_r rd_r wr_r} {

  global w_fi_flist w_fi_w_fi_flistbox w_fi_lbframe

  set color black

  if {$rd == 1} {
    set props "Read "
    set img_r light_green
  } elseif {$rd == 0} {
    set props "     "
    if {$rd_r} {
      set img_r light_yellow
    } else {
      set img_r light_red
    }
  } else {
    set props "     "
    set img_r not_allowed
  }

  if {$wr == 1} {
    set props "$props  Write "
    set img_w light_green
  } elseif {$wr == 0} {
    set props "$props        "
    if {$wr_r} {
      set img_w light_yellow
    } else {
      set img_w light_red
    }
  } else {
    set props "     "
    set img_w not_allowed
  }

  if {$ctrl} {
    set props "$props Control "
    set img_c light_green
  } else {
    set props "$props         "
    if {$ctrl_r} {
      set img_c light_yellow
    } else {
      set img_c light_red
    }
  }


  set name [string range "$name                    " 0 15]

  #this must be done bij deleting & inserting.
  set lnr [lsearch -glob $w_fi_flist $f]
  if {$lnr >= 0} {
    $w_fi_w_fi_flistbox delete $lnr
    $w_fi_w_fi_flistbox insert $lnr "$name
"
   # geen $props
#    $w_fi_lbframe.lr$lnr configure -image $img_r
#    $w_fi_lbframe.lw$lnr configure -image $img_w
#    $w_fi_lbframe.lc$lnr configure -image $img_c
#Bounded windows in lists must be created again
    fancylabel $w_fi_w_fi_flistbox.lr$lnr -image $img_r
    fancylabel $w_fi_w_fi_flistbox.lw$lnr -image $img_w
    fancylabel $w_fi_w_fi_flistbox.lc$lnr -image $img_c

    $w_fi_w_fi_flistbox window create $lnr.44 -window
$w_fi_w_fi_flistbox.lc$lnr -pady 3
    $w_fi_w_fi_flistbox window create $lnr.50 -window
$w_fi_w_fi_flistbox.lw$lnr -pady 3
    $w_fi_w_fi_flistbox window create $lnr.57 -window
$w_fi_w_fi_flistbox.lr$lnr -pady 3

   }
   #change color

   $w_fi_w_fi_flistbox item configure $lnr -foreground $color

}

set Name $Parent.f.f183
# Procedures----------------------------

set Name $Parent.f.f188
# Procedures----------------------------

set Name $Parent.f.f190
# Procedures----------------------------

set Name $Parent.f.l194
# Procedures----------------------------

set Name $Parent.f.l196
# Procedures----------------------------

set Name $Parent.f.l198
# Procedures----------------------------

set Name $Parent.f.l200
# Procedures----------------------------

set Name $Parent.b182
# Procedures----------------------------
#- TOP LEVEL procedures----------------------------------

set Name .start2
set Parent $Name
global Procs
set Procs($Name) { w_st2:init w_st2:start_conference w_st2:show
w_st2:hide w_st2:define_m_menu w_st2:show_m_menu {bind .start2
<Alt-s>} {bind .start2 <Alt-q>} {bind .start2 <KeyPress-
Return>} {bind .start2 <Alt-i>} {bind .start2 <Alt-m>}}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) start2:
set Glob_prefix($Name) start2_
proc w_st2:init {no_iscreen m_nr u_nr no_c u_atm dbg vc vs vp
sp sl_emu} {

  global wst2_startpressed
  global wst2_no_camera wst2_coding wst2_half_dupl
  global wst2_use_atm wst2_enable_debug
  global wst2_meetingnr_inp wst2_usernr_inp
  global wst2_meeting_menu
  global wst2_vo_size wst2_vo_preview
  global wst2_pict
  global wst2_sl_emu

  set wst2_meeting_menu ".start2.m_inp"
  set wst2_startpressed 0

  set wst2_coding $vc
  set wst2_no_camera $no_c
  set wst2_half_dupl [sy:is_windows]
```

```
  set wst2_vo_size $vs
  set wst2_vo_preview $vp
  set wst2_pict $sp

  set wst2_use_atm $u_atm

  set wst2_meetingnr_inp $m_nr
  set wst2_usernr_inp $u_nr

  set wst2_enable_debug $dbg
  set wst2_sl_emu $sl_emu

  #read user-database so names can be mapped to numbers
  ud:read_user_database

  if {$no_iscreen} {
    w_st2:start_conference 1
  } else {
    w_st2:show
  }

  main:status_message "READY"
}
proc w_st2:start_conference no_iscreen {

  global wst2_enable_debug wst2_use_atm wst2_no_camera
wst2_meetingnr_inp wst2_usernr_inp wst2_coding
  global wst2_vo_size wst2_vo_preview wst2_half_dupl
  global wst2_pict
  global wst2_sl_emu

  main:status_message "Starting conference"

  catch {if {$wst2_startpressed} {return}}
  set wst2_startpressed 1

  if {$wst2_use_atm} {
     set a_adr  [sy:get_atm_adr [sy:get_local_adr]]
   } else {
      set a_adr ""
   }

   #convert to defined number
   set unr [ud:get_usernumber $wst2_usernr_inp]
   set u_name [ud:get_username $wst2_usernr_inp 0]

  #now convert to complete name
  if {$unr >= 0} {
    set u_name [ud:get_username $unr 0]
  } else {
    set u_name $wst2_usernr_inp
  }
  ud:set_current_username $u_name $unr

  #this name will be passed through to other participants
  set u_id $u_name

  #:::This is for passing numbers instead of names....
  #user unknown: name can be passed through
  #if {$unr == -1} {
  #  set u_id $wst2_usernr_inp
  #} else {
  #  set u_id $unr
  #}

   set u_id $wst2_usernr_inp

   #initialize everything and start network layer

   #supercontrol
   w_sc:init_win $wst2_enable_debug

   #Local multimedia
   set audio_fail [sy:init_audio]
   nl:init_vid_coding $wst2_coding
   nl:set_vidr_auto 1

   lm:init_win $wst2_half_dupl $wst2_vo_size $wst2_vo_preview

   #get reflector address
   set refl_adr [fr:get_reflector $wst2_meetingnr_inp]

   if {[sy:check_address $refl_adr $a_adr $wst2_sl_emu]} {

    #all is well !

    nl:start_network $u_name $refl_adr $a_adr $wst2_sl_emu

   #how to end DTC-session
   if {!$no_iscreen} {
     w_fc:first_initialisation "w_st2:show" 0
   } else {
     w_fc:first_initialisation "sy:quit_all" $wst2_enable_debug
   }

   #initialise floors
   set succeed [fr:read_meetingdbase $unr $wst2_meetingnr_inp]

   if {!$succeed} {
     nl:stop_netlayer
     w_fc:kill_windows
     return
   }

   #start DTC
   w_fc:start $wst2_pict

   if {!$no_iscreen} {
     w_st2:hide
   }
   set wst2_startpressed 0

   #check for failures and enable floor-reception

   if {$audio_fail} {
     fu:audio_unusable
   }
   if {$wst2_no_camera} {
     fu:video_write_unusable
   }

   foreach f [fd:all_floor_ids] {
     fu:clear_mute W $f
     fu:clear_mute R $f
   }
   lm:show_muting

   nl:enable_reception
   main:status_message "READY"
  }


}
proc w_st2:show {} {

  wincreate .start2
  w_st2:define_m_menu
  wshow .start2
  focus .start2
  focus .start2.e_nr

}
proc w_st2:hide {} {

  windestroy .start2

}
proc w_st2:define_m_menu {} {

  global wst2_meeting_menu

  set mlist [fr:get_all_meetings_info]

  for {set t 0} {$t < [llength $mlist]} {incr t 2} {
    set m_nr [lindex $mlist $t]
    set m_name [lindex $mlist [expr $t + 1]]
    lappend menu "\{$m_name \($m_nr\)\} command \{set
wst2_meetingnr_inp $m_nr\}"
  }

  set Menu_string($wst2_meeting_menu) "\{$menu\} -background
white -activebackground black -activeforeground white -tearoff
0"

 #make menu itself

 eval "make_menu $wst2_meeting_menu
$Menu_string($wst2_meeting_menu)"

}
proc w_st2:show_m_menu {} {

  global wst2_meeting_menu

  update idletasks
  set y [expr [winfo rooty "$wst2_meeting_menu"] + [winfo
height "$wst2_meeting_menu"]]
  set x [winfo rootx "$wst2_meeting_menu"]

  tk_popup $wst2_meeting_menu.m $x $y
}

set Name $Parent.l178
# Procedures----------------------------

set Name $Parent.l180
# Procedures----------------------------

set Name $Parent.l182
# Procedures----------------------------

set Name $Parent.l184
# Procedures----------------------------

set Name $Parent.m_inp
# Procedures----------------------------
global Procs
set Procs($Name) { }

set Name $Parent.b_start
# Procedures----------------------------
global Procs
set Procs($Name) { }
global Proc_prefix Glob_prefix
set Proc_prefix($Name) start2:b_start:
set Glob_prefix($Name) start2_b_start_

set Name $Parent.b_quit
# Procedures----------------------------
global Procs
set Procs($Name) {{bind .start2.b_quit <KeyPress-Return>}}

set Name $Parent.e_nr
# Procedures----------------------------
global Procs
set Procs($Name) { }
global Proc_prefix Glob_prefix
set Proc_prefix($Name) start2:e_nr:
set Glob_prefix($Name) start2_e_nr_

set Name $Parent.b_mlist
# Procedures----------------------------
global Procs
set Procs($Name) { {bind .start2.b_mlist <KeyPress-Return>}}
```

```
#- TOP LEVEL procedures---------------------------------

set Name .whiteboard
set Parent $Name
global Procs
set Procs($Name) {wb:MM wb:PA wb:Segs wb:boxDraw wb:boxEnd
wb:canvasBindings wb:canvasDrag wb:canvasMark wb:click_delete
wb:delete wb:drawEnd wb:drawTo wb:eraseDraw wb:eraseEnd wb:init
wb:itemEndGrab wb:itemGrab wb:itemMove wb:lineDraw wb:lineEnd
wb:moveTo wb:ovalDraw wb:ovalEnd wb:setBoxMode wb:setEraseMode
wb:setLineMode wb:setMode wb:setOvalMode wb:setStrokeMode
wb:setTextMode wb:showMode wb:showScrollRegion
wb:showScrollRegionOld wb:startGrab wb:strokeDraw wb:strokeEnd
wb:textDraw wb:textEnd wb:whatsThere wb:feedback
wb:erase_selection wb:show wb:hide wb:destroy}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) wb:
set Glob_prefix($Name) wb_
proc wb:MM {} {
    wb:PA Move
}
proc wb:PA array {

    puts stdout $array
    foreach name [lsort [array names $array]] {
          set ref [format "%s(%s)" $array $name]
          set item [eval "set $ref"]
          puts stdout [format "%8s %s" $name $item]
    }
}
proc wb:Segs {} {
    global wb_N wb_Segments
    for {set i 0} {$i < $wb_N} {incr i} {
          puts stdout [format "%s %s" $i $wb_Segments($i)]
    }
}
proc wb:boxDraw {can x y} {
    global wb_X wb_Y wb_N wb_Segments wb_itemN
    if {$wb_N > 0} {
          $can delete $wb_Segments([expr $wb_N-1])
    }
    return [$can create rectangle $wb_X(0) $wb_Y(0) $x $y -tags
box$wb_itemN]
}
proc wb:boxEnd {can x y} {
    return [wb:drawTo $can $x $y wb:boxDraw]
}
proc wb:canvasBindings can {
# Keybindings to make strokes (Left button)
    bind $can <Button-1> {wb:moveTo %W %x %y}
    bind $can <B1-Motion> {wb:drawTo %W %x %y}
    bind $can <ButtonRelease-1> {wb:drawEnd %W %x %y}

# Keybindings to delete strokes (control-Left)
    bind $can <Control-Button-1> {wb:click_delete %W %x %y}

# Keybindings to scroll (Middle button, fast or slow drag via
Shift)
    bind $can <Button-2> {wb:canvasMark %W %x %y }
    bind $can <B2-Motion> {wb:canvasDrag %W %x %y slow }
    bind $can <Shift-Button-2> {wb:canvasMark %W %x %y }
    bind $can <Shift-B2-Motion> {wb:canvasDrag %W %x %y fast }
  #for 2 mouse buttons
    bind $can <Control-Button-1> {wb:canvasMark %W %x %y }
    bind $can <Control-B1-Motion> {wb:canvasDrag %W %x %y slow
}
    bind $can <Alt-Button-1> {wb:canvasMark %W %x %y }
    bind $can <Alt-B1-Motion> {wb:canvasDrag %W %x %y fast }

# Keybindings to query the canvas (Shift, left or right to
determine range)
    bind $can <Shift-Button-1> { wb:feedback [wb:whatsThere %W
%x %y 1] }
    bind $can <Shift-Button-3> { wb:feedback [wb:whatsThere %W
%x %y 5] }

# Keybindings to move items (Button 3)
    bind $can <Button-3> { wb:itemGrab %W %x %y }
    bind $can <B3-Motion> { wb:itemMove %W %x %y }
    bind $can <ButtonRelease-3> { wb:itemEndGrab %W %x %y }
}
proc wb:canvasDrag {can x y speed} {
    global wb_canMarkX
    if {$speed == "slow"} {
          set x [expr $wb_canMarkX+($x-$wb_canMarkX)/10]
    }
    $can scan dragto $x $y
}
proc wb:canvasMark {can x y} {
    global wb_canMarkX
    set wb_canMarkX $x
    $can scan mark $x $y
}
proc wb:click_delete {w x y} {

  #delete without changing mode

  global wb_mode

  set old_mode $wb_mode
  wb:setEraseMode $can
  wb:delete $w $x $y
  set mode $old_mode

}
proc wb:delete {can x y {halo 2}} {
    global wb_canBackground

    set cx [$can canvasx $x]
    set cy [$can canvasy $y]

    set item [$can find closest $cx $cy $halo]
    if {($item != {}) && ($item != $wb_canBackground)} {
          $can delete $item
    }
}
proc wb:drawEnd {can x y {proc default}} {
    global wb_endProc wb_itemN wb_N
    if {$proc == "default"} {
          set proc $wb_endProc
    }
    set item [$proc $can $x $y]
    incr wb_itemN
    set wb_N -1
    return $item
}
proc wb:drawTo {can x y {proc default}} {
    global wb_drawProc
    global wb_X wb_Y wb_N wb_Segments
    global wb_Move
    if {$wb_Move(item) == "SEL"} {
                #clear selection
          $can delete $wb_Move(lastBox)
                set wb_Move(items) {}
          set wb_Move(item) {}
                $can dtag SEL
    }
    if {![info exists wb_N] || $wb_N < 0} {
                wb:feedback ""
          return
    }
    if {$proc == "default"} {
          set proc $wb_drawProc
    }
    set x [$can canvasx $x]
    set y [$can canvasy $y]
    set item [$proc $can $x $y]
    set wb_Segments($wb_N) $item
    incr wb_N
    set wb_X($wb_N) $x
    set wb_Y($wb_N) $y
    return $item
}
proc wb:eraseDraw {can x y} {
    global wb_X wb_Y wb_N wb_Segments wb_itemN
    set items [$can find overlapping $wb_X($wb_N) $wb_Y($wb_N)
$x $y]
    foreach item $items {
          global wb_canBackground
          if {$item != $wb_canBackground} {
              $can delete $item
          }
    }
    return {}
}
proc wb:eraseEnd {can x y} {
    wb:drawTo $can $x $y wb:eraseDraw
    return {}
}
proc wb:init {} {

  global wb_itemN wb_tnum wb_Move wb_can wb_feedback
wb_canWidth wb_canHeight wb_win

  set wb_win .whiteboard

  wincreate $wb_win
  whide $wb_win

  set wb_feedback ""

  set wb_itemN 0
  set wb_tnum 0
  set wb_Move(item) {}
  set wb_can "$wb_win.fr.can"

  set wb_canWidth 600
  set wb_canHeight 400

  $wb_can configure -scrollregion "0 0 $wb_canWidth
$wb_canHeight"

  wb:canvasBindings $wb_can
  wb:setStrokeMode $wb_can

  wb:showScrollRegion $wb_can

}
proc wb:itemEndGrab {can x y} {
    global wb_Move
    if {![info exists wb_Move(item)] || $wb_Move(item) !=
"DRAG"} {
          return
    }
    if {$x > $wb_Move(X0)} {
          set wb_Move(X1) $x
    } else {
          set wb_Move(X1) $wb_Move(X0)
          set wb_Move(X0) $x
    }
    if {$y > $wb_Move(Y0)} {
          set wb_Move(Y1) $y
    } else {
          set wb_Move(Y1) $wb_Move(Y0)
          set wb_Move(Y0) $y
    }
    set wb_Move(items) [$can find enclosed $wb_Move(X0)
$wb_Move(Y0) $wb_Move(X1) $wb_Move(Y1)]
    set wb_Move(lastBox) [wb:drawEnd $can $x $y wb:boxEnd]
    if {$wb_Move(items) == {}} {
          set wb_Move(item) {}
          $can delete $wb_Move(lastBox)
                $can dtag SEL
```

```
          wb:feedback "Nothing in selection box"
    } else {
          $can itemconfigure $wb_Move(lastBox) -tags SEL
          $can addtag SEL enclosed $wb_Move(X0) $wb_Move(Y0)
$wb_Move(X1) $wb_Move(Y1)
          set wb_Move(item) SEL
          wb:feedback "Move items $wb_Move(items)"
    }
}
proc wb:itemGrab {can x y} {
    global wb_Move wb_canBackground
    case $wb_Move(item) {
          default {
              # Normal case, click to select or initiate area
select
              wb:startGrab $can $x $y
          }
          SEL {
              if {(($wb_Move(X0) < $wb_Move(X1)) &&  (($x <
$wb_Move(X0)) || ($x > $wb_Move(X1)))) ||  (($wb_Move(X0) >
$wb_Move(X1)) &&  (($x < $wb_Move(X1)) || ($x > $wb_Move(X0))))
||  (($wb_Move(Y0) < $wb_Move(Y1)) &&  (($y < $wb_Move(Y0)) ||
($y > $wb_Move(Y1)))) ||  (($wb_Move(Y0) > $wb_Move(Y1)) &&
(($y < $wb_Move(Y1)) || ($y > $wb_Move(Y0))))} {
                    $can delete $wb_Move(lastBox)
                    wb:startGrab $can $x $y
              } else {
                    set wb_Move(X) $x
                    set wb_Move(Y) $y
              }
          }
    }
}
proc wb:itemMove {can x y} {
    global wb_Move
    if {$wb_Move(item) == {}} {
          return
    }
    if {$wb_Move(item) == "DRAG"} {
          # Dragging out a selection
          wb:drawTo $can $x $y wb:boxDraw
          set wb_Move(X1) $x
          set wb_Move(Y1) $y
    } else {
          set dx [expr $x-$wb_Move(X)]
          set dy [expr $y-$wb_Move(Y)]
          $can move $wb_Move(item) $dx $dy
          incr wb_Move(X0) $dx
          incr wb_Move(X1) $dx
          incr wb_Move(Y0) $dy
          incr wb_Move(Y1) $dy
          set wb_Move(X) $x
          set wb_Move(Y) $y
    }
}
proc wb:lineDraw {can x y} {
    global wb_X wb_Y wb_N wb_Segments wb_itemN
    if {$wb_N > 0} {
          $can delete $wb_Segments([expr $wb_N-1])
    }
    return [$can create line $wb_X(0) $wb_Y(0) $x $y -tags
line$wb_itemN]
}
proc wb:lineEnd {can x y} {
    global wb_itemN
    global wb_X wb_Y
    return [wb:drawTo $can $x $y wb:lineDraw]
}
proc wb:moveTo {can x y} {

   #IInitialise positions and start draw
    global wb_Segments
    global wb_X wb_Y wb_N

   set wb_X(0) [$can canvasx $x]
    set wb_Y(0) [$can canvasy $y]
    set wb_N 0
    set wb_Segments(0) {}
}
proc wb:ovalDraw {can x y} {
    global wb_X wb_Y wb_N wb_Segments wb_itemN
    if {$wb_N > 0} {
          $can delete $wb_Segments([expr $wb_N-1])
    }
    return [$can create oval $wb_X(0) $wb_Y(0) $x $y -tags
oval$wb_itemN]
}
proc wb:ovalEnd {can x y} {
    return [wb:drawTo $can $x $y wb:ovalDraw]
}
proc wb:setBoxMode can {
    global wb_drawProc wb_endProc wb_mode
    set wb_mode stroke
    set wb_drawProc wb:boxDraw
    set wb_endProc wb:boxEnd
    wb:showMode stroke
    $can configure -cursor cross
}
proc wb:setEraseMode can {
    global wb_drawProc wb_endProc wb_mode
    set wb_mode stroke
    set wb_drawProc wb:eraseDraw
    set wb_endProc wb:eraseEnd
    wb:showMode stroke
    $can configure -cursor dotbox
}
proc wb:setLineMode can {
    global wb_drawProc wb_endProc wb_mode
    set wb_mode stroke
    set wb_drawProc wb:lineDraw
    set wb_endProc wb:lineEnd
    wb:showMode stroke
    $can configure -cursor crosshair
}
proc wb:setMode {can newMode} {
    # can is ignored, but we should have per-canvas state
    global wb_mode wb_drawProc wb_endProc
    case $newMode {
          stroke {
              wb:setStrokeMode $can
          }
          line {
              wb:setLineMode $can
          }
          box {
              wb:setBoxMode $can
          }
          oval {
              wb:setOvalMode $can
          }
          text {
              wb:setTextMode $can
          }
          erase {
              wb:setEraseMode $can
          }
          default {
              wb:feedback "Unsupported mode: $newMode"
              return
          }
    }
    set mode $newMode
    wb:showMode $mode
}
proc wb:setOvalMode can {
    global wb_drawProc wb_endProc wb_mode
    set wb_mode stroke
    set wb_drawProc wb:ovalDraw
    set wb_endProc wb:ovalEnd
    wb:showMode stroke
    $can configure -cursor target
}
proc wb:setStrokeMode can {
    global wb_drawProc wb_endProc wb_mode
    set wb_mode stroke
    set wb_drawProc wb:strokeDraw
    set wb_endProc wb:strokeEnd
    wb:showMode stroke
    $can configure -cursor pencil
}
proc wb:setTextMode can {
    global wb_drawProc wb_endProc wb_mode
    set wb_mode stroke
    set wb_drawProc wb:textDraw
    set wb_endProc wb:textEnd
    wb:showMode text
    $can configure -cursor tcross
}
proc wb:showMode mode {

    global wb_showmode
    set wb_showmode $mode
}
proc wb:showScrollRegion can {
    global wb_canWidth wb_canHeight wb_canBackground
    if [info exists wb_canBackground] {
          $can delete $wb_canBackground
    }
    set wb_canBackground [$can create rectangle 0 0
$wb_canWidth $wb_canHeight  -fill white -outline {}]
    $can lower $wb_canBackground
}
proc wb:showScrollRegionOld can {
    global wb_canWidth wb_canHeight wb_canBackground
    set items [$can find withtag scrollBorder]
    foreach item $items {
          $can delete $item
    }
    set x1 3 ; set y1 3
    set x2 [expr $canWidth-3]
    set y2 [expr $canHeight-3]
    $can create line $x1 $y1 $x2 $y1 -stipple gray50 -tags
{scrollBorder}
    $can create line $x2 $y1 $x2 $y2 -stipple gray50 -tags
{scrollBorder}
    $can create line $x2 $y2 $x1 $y2 -stipple gray50 -tags
{scrollBorder}
    $can create line $x1 $y2 $x1 $y1 -stipple gray50 -tags
{scrollBorder}
}
proc wb:startGrab {can x y} {
    global wb_Move wb_canBackground
    set halo 5
    $can dtag SEL
    set item [$can find closest $x $y $halo]
    if {$item != {} && $item != $wb_canBackground} {
          wb:feedback "Selected item $item [$can gettags
$item]"
          set wb_Move(item) $item
          set wb_Move(X) $x
          set wb_Move(Y) $y
          set wb_Move(X0) $x
          set wb_Move(Y0) $y
          set wb_Move(X1) $x
          set wb_Move(Y1) $y
    } else {
          wb:feedback "Selecting a region"
          set wb_Move(X0) $x
          set wb_Move(Y0) $y
          set wb_Move(item) DRAG
          wb:moveTo $can $x $y
    }
}
proc wb:strokeDraw {can x y} {
```

```
    global wb_X wb_Y wb_N
    return [$can create line $wb_X($wb_N) $wb_Y($wb_N) $x $y -
tags {line}]
}
proc wb:strokeEnd {can x y} {
    global wb_itemN
    wb:drawTo $can $x $y
    # Replace straight-line segments with bezier curve
    global wb_X wb_Y wb_N wb_Segments
    set createCmd "$can create line"
    for {set i 0} {$i <= $wb_N} {incr i} {
          lappend createCmd $wb_X($i) $wb_Y($i)
          if {$i != $wb_N} {
              $can delete $wb_Segments($i)
          }
    }
    lappend createCmd -joinstyle round -smooth 1 -tags
stroke$wb_itemN
    return [eval $createCmd]
}
proc wb:textDraw {can x y} {
    global wb_X wb_Y wb_N wb_Segments wb_itemN
    if {$wb_N > 0} {
          $can delete $wb_Segments([expr $wb_N-1])
    }
    return [$can create rectangle $wb_X(0) $wb_Y(0) $x $y -tags
textBox$wb_itemN]
}
proc wb:textEnd {can x y} {
    global wb_Segments wb_X wb_Y wb_N wb_itemN
    set width [expr {($wb_X($wb_N) > $wb_X(0)) ? ($wb_X($wb_N)
- $wb_X(0)) : ($wb_X(0) - $wb_X($wb_N))}]
    set height [expr {($wb_Y($wb_N) > $wb_Y(0)) ? ($wb_Y($wb_N)
- $wb_Y(0)) : ($wb_Y(0) - $wb_Y($wb_N))}]
    if {$width !=0 && $height != 0} {
    wb:drawTo $can $x $y wb:textDraw

    global wb_tnum
    incr wb_tnum
    text $can.text$wb_tnum
    $can delete textBox$wb_itemN
    return [$can create window $wb_X(0) $wb_Y(0) -window
$can.text$wb_tnum  -width $width -height $height -anchor nw -
tags textBox$wb_itemN]
  } else {
  return ""
 }
}
proc wb:whatsThere {can x y halo} {
    set cx [$can canvasx $x]
    set cy [$can canvasy $y]

    set things [$can find overlapping [expr $cx-$halo] [expr
$cy-$halo]  [expr $cx+$halo] [expr $cy+$halo]]

    set lasttag {}
    set answer "${things}:"
    foreach item $things {
          set tags [$can gettags $item]
          foreach tag $tags {
              if {$lasttag == $tag} {
                    continue
              }
              if {($tag != "Scale") && ($tag != "current")} {
                    append answer "$tag "
              }
              set lasttag $tag
          }
    }
    return $answer
}
proc wb:feedback par {

  global wb_feedback
  set wb_feedback $par

}
proc wb:erase_selection can {

  global wb_Move

  $can delete $wb_Move(item)

  #clear selection
  catch {$can delete $wb_Move(lastBox)}
  set wb_Move(items) {}
   set wb_Move(item) {}
   $can dtag SEL
   return {}
}
proc wb:show {} {

  global wb_win

  wshow $wb_win
}
proc wb:hide {} {

  global wb_win

  whide $wb_win

}
proc wb:destroy {} {

  global wb_win

  catch {windestroy $wb_win}

}

set Name $Parent.fr
# Procedures----------------------------

set Name $Parent.fr.can
# Procedures----------------------------

set Name $Parent.fr.sb26
# Procedures----------------------------

set Name $Parent.fr.sb28
# Procedures----------------------------

set Name $Parent.f_st
# Procedures----------------------------

set Name $Parent.f_st.l19
# Procedures----------------------------

set Name $Parent.f_st.l30
# Procedures----------------------------

set Name $Parent.f_st.l22
# Procedures----------------------------

set Name $Parent.f_st.l27
# Procedures----------------------------

set Name $Parent.f_st.l23
# Procedures----------------------------

set Name $Parent.f_st.l25
# Procedures----------------------------

set Name $Parent.f_st.b14
# Procedures----------------------------

set Name $Parent.f_bt
# Procedures----------------------------

set Name $Parent.f_bt.b11
# Procedures----------------------------

set Name $Parent.f_bt.b12
# Procedures----------------------------

set Name $Parent.f_bt.b13
# Procedures----------------------------

set Name $Parent.f_bt.b18
# Procedures----------------------------

set Name $Parent.f_bt.b10
# Procedures----------------------------

set Name $Parent.f_bt.b15
# Procedures----------------------------
#- TOP LEVEL procedures---------------------------------

set Name .w_sle
set Parent $Name
global Procs
set Procs($Name) {w_sle:adding w_sle:deleting
w_sle:FP_Tokenplease_ind w_sle:FC_Reject_ind
w_sle:FP_Withdraw_ind w_sle:Tokengive_ind
w_sle:FC_Tokengrab_conf w_sle:FC_Tokengrab_ind
w_sle:text_stream_ind w_sle:show w_sle:hide w_sle:FC_Reset_ind}
global Proc_prefix Glob_prefix
set Proc_prefix($Name) w_sle:
set Glob_prefix($Name) w_sle_
proc w_sle:adding {} {

    global w_sle_ipnr w_sle_name w_sle_floors w_sle_text

    set inp "adding $w_sle_ipnr $w_sle_name"
    w_sc:addlst_pipe_in "EMU: $inp"
    nl:eval_incoming $inp

}
proc w_sle:deleting {} {

    global w_sle_ipnr w_sle_name w_sle_floors w_sle_text

    set inp "deleting $w_sle_ipnr"
    w_sc:addlst_pipe_in "EMU: $inp"
    nl:eval_incoming $inp

}
proc w_sle:FP_Tokenplease_ind {} {

     global w_sle_ipnr w_sle_name w_sle_floors w_sle_text

    set inp "FP_Tokenplease_ind  $w_sle_floors $w_sle_ipnr
$w_sle_text"
    w_sc:addlst_pipe_in "EMU: $inp"
    nl:eval_incoming $inp

}
proc w_sle:FC_Reject_ind {} {

    global w_sle_ipnr w_sle_name w_sle_floors w_sle_text

    set inp "FC_Reject_ind  $w_sle_floors $w_sle_ipnr
$w_sle_text"
    w_sc:addlst_pipe_in "EMU: $inp"
    nl:eval_incoming $inp

}
proc w_sle:FP_Withdraw_ind {} {

    global w_sle_ipnr w_sle_name w_sle_floors w_sle_text

    set inp "FP_Withdraw_ind   $w_sle_floors $w_sle_ipnr"
    w_sc:addlst_pipe_in "EMU: $inp"
```

```
    nl:eval_incoming $inp

}
proc w_sle:Tokengive_ind {} {

    global w_sle_ipnr w_sle_name w_sle_floors w_sle_text

    set inp "Tokengive_ind  $w_sle_floors $w_sle_ipnr"
    w_sc:addlst_pipe_in "EMU: $inp"
    nl:eval_incoming $inp
}
proc w_sle:FC_Tokengrab_conf {} {

   global w_sle_ipnr w_sle_name w_sle_floors w_sle_text

    set inp "FC_Tokengrab_conf  $w_sle_floors $w_sle_ipnr"
    w_sc:addlst_pipe_in "EMU: $inp"
    nl:eval_incoming $inp
}
proc w_sle:FC_Tokengrab_ind {} {

    global w_sle_ipnr w_sle_name w_sle_floors w_sle_text

    set inp "FC_Tokengrab_ind  $w_sle_floors"
    w_sc:addlst_pipe_in "EMU: $inp"
    nl:eval_incoming $inp
}
proc w_sle:text_stream_ind {} {

    global w_sle_ipnr w_sle_name w_sle_floors w_sle_text
w_sle_channel

    set inp "text_stream_ind $w_sle_channel $w_sle_ipnr
$w_sle_text"
    w_sc:addlst_pipe_in "EMU: $inp"
    nl:eval_incoming $inp

}
proc w_sle:show {} {

   global w_sle_ipnr w_sle_name w_sle_floors w_sle_text
w_sle_channel

   set w_sle_ipnr "130.89.100.100"
   set w_sle_name "2"
   set w_sle_floors "WV1WA1WT1"
   set w_sle_text "Test"
   set w_sle_channel "1"

   wincreate .w_sle

}
proc w_sle:hide {} {

   catch {windestroy .w_sle}

}
proc w_sle:FC_Reset_ind {} {

    global w_sle_ipnr w_sle_name w_sle_floors w_sle_text

    set inp "FC_Reset_ind  $w_sle_floors $w_sle_ipnr"
    w_sc:addlst_pipe_in "EMU: $inp"
    nl:eval_incoming $inp

}

set Name $Parent.l237
# Procedures----------------------------

set Name $Parent.l238
# Procedures----------------------------

set Name $Parent.l239
# Procedures----------------------------

set Name $Parent.e240
# Procedures----------------------------

set Name $Parent.e241
# Procedures----------------------------

set Name $Parent.b242
# Procedures----------------------------

set Name $Parent.b243
# Procedures----------------------------

set Name $Parent.b244
# Procedures----------------------------

set Name $Parent.b245
# Procedures----------------------------

set Name $Parent.b246
# Procedures----------------------------

set Name $Parent.b247
# Procedures----------------------------

set Name $Parent.b248
# Procedures----------------------------

set Name $Parent.b249
# Procedures----------------------------

set Name $Parent.e250
# Procedures----------------------------

set Name $Parent.l252
# Procedures----------------------------

set Name $Parent.b253
# Procedures----------------------------

set Name $Parent.e255
# Procedures----------------------------

set Name $Parent.b232
# Procedures----------------------------

set Name $Parent.e233
# Procedures----------------------------

set Name $Parent.l234
# Procedures----------------------------
proc mainwincr_system {nr} {
set w [win_name .system $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -borderwidth 2 -relief raised
set_onkill $w {destroy $w}
wm protocol $w WM_DELETE_WINDOW "destroy $w"
if {$nr > 0} {
wm title $w "system_${nr}"
} else {wm title $w "system"}
set Toppos($w) 1
set Topgeom($w) 1
set stretchX($w) 1
set stretchY($w) 1
set Topmin($w) 1
set TopIsModule($w) 1
global is_guibuilder
wm geometry $w 133x64
wm geometry $w +401+126
wm resizable $w 1 1
wm minsize $w 133 64

set Name $w
set Parent $Name
}
proc mainwincr_network_layer {nr} {
set w [win_name .network_layer $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -borderwidth 2 -relief raised
set_onkill $w {destroy $w}
wm protocol $w WM_DELETE_WINDOW "destroy $w"
if {$nr > 0} {
wm title $w "network_layer_${nr}"
} else {wm title $w "network_layer"}
set Toppos($w) 1
set Topgeom($w) 1
set stretchX($w) 1
set stretchY($w) 1
set Topmin($w) 1
set TopIsModule($w) 1
global is_guibuilder
wm geometry $w 139x66
wm geometry $w +353+90
wm resizable $w 1 1
wm minsize $w 139 66

set Name $w
set Parent $Name
}
proc mainwincr_users_db {nr} {
set w [win_name .users_db $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -borderwidth 2 -relief raised
set_onkill $w {destroy $w}
wm protocol $w WM_DELETE_WINDOW "destroy $w"
if {$nr > 0} {
wm title $w "users_db_${nr}"
} else {wm title $w "users_db"}
set Toppos($w) 1
set Topgeom($w) 1
set stretchX($w) 1
set stretchY($w) 1
set Topmin($w) 1
set TopIsModule($w) 1
global is_guibuilder
wm geometry $w 139x62
wm geometry $w +324+65
wm resizable $w 1 1
wm minsize $w 139 62

set Name $w
set Parent $Name
}
proc mainwincr_floor_req_grant_db {nr} {
set w [win_name .floor_req_grant_db $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
```

```
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -borderwidth 2 -relief raised
set_onkill $w {destroy $w}
wm protocol $w WM_DELETE_WINDOW "destroy $w"
if {$nr > 0} {
wm title $w "floor_req_grant_db_${nr}"
} else {wm title $w "floor_req_grant_db"}
set Toppos($w) 1
set Topgeom($w) 1
set stretchX($w) 1
set stretchY($w) 1
set Topmin($w) 1
set TopIsModule($w) 1
global is_guibuilder
wm geometry $w 167x57
wm geometry $w +423+169
wm resizable $w 1 1
wm minsize $w 167 57

set Name $w
set Parent $Name
}
proc mainwincr_floor_control {nr} {
set w [win_name .floor_control $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -borderwidth 2 -relief raised
set_onkill $w {destroy $w}
wm protocol $w WM_DELETE_WINDOW "destroy $w"
if {$nr > 0} {
wm title $w "floor_control_${nr}"
} else {wm title $w "floor_control"}
set Toppos($w) 1
set Topgeom($w) 1
set stretchX($w) 1
set stretchY($w) 1
set Topmin($w) 1
set TopIsModule($w) 1
global is_guibuilder
wm geometry $w 140x58
wm geometry $w +117+150
wm resizable $w 1 1
wm minsize $w 140 58

set Name $w
set Parent $Name
}
proc mainwincr_main {nr} {
set w [win_name .main $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -borderwidth 2 -relief raised
set_onkill $w {destroy $w}
wm protocol $w WM_DELETE_WINDOW "destroy $w"
if {$nr > 0} {
wm title $w "main_${nr}"
} else {wm title $w "main"}
set Toppos($w) 1
set Topgeom($w) 1
set stretchX($w) 1
set stretchY($w) 1
set Topmin($w) 1
set TopIsModule($w) 1
global is_guibuilder
wm geometry $w 132x44
wm geometry $w +93+123
wm resizable $w 1 1
wm minsize $w 132 44

set Name $w
set Parent $Name
}
proc mainwincr_floor_use {nr} {
set w [win_name .floor_use $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -borderwidth 2 -relief raised
set_onkill $w {destroy $w}
wm protocol $w WM_DELETE_WINDOW "destroy $w"
if {$nr > 0} {
wm title $w "floor_use_${nr}"
} else {wm title $w "floor_use"}
set Toppos($w) 1
set Topgeom($w) 1
set stretchX($w) 1
set stretchY($w) 1
set Topmin($w) 1
set TopIsModule($w) 1
global is_guibuilder
wm geometry $w 143x49
wm geometry $w +62+88
wm resizable $w 1 1
wm minsize $w 143 49

set Name $w
set Parent $Name
}
proc mainwincr_floor_def {nr} {
set w [win_name .floor_def $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -borderwidth 2 -relief raised
set_onkill $w {destroy $w}
wm protocol $w WM_DELETE_WINDOW "destroy $w"
if {$nr > 0} {
wm title $w "floor_def_${nr}"
} else {wm title $w "floor_def"}
set Toppos($w) 1
set Topgeom($w) 1
set stretchX($w) 1
set stretchY($w) 1
set Topmin($w) 1
set TopIsModule($w) 1
global is_guibuilder
wm geometry $w 140x49
wm geometry $w +39+56
wm resizable $w 1 1
wm minsize $w 140 49

set Name $w
set Parent $Name
}
proc mainwincr_floor_read_db {nr} {
set w [win_name .floor_read_db $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -borderwidth 2 -relief raised
set_onkill $w {destroy $w}
wm protocol $w WM_DELETE_WINDOW "destroy $w"
if {$nr > 0} {
wm title $w "floor_read_db_${nr}"
} else {wm title $w "floor_read_db"}
set Toppos($w) 1
set Topgeom($w) 1
set stretchX($w) 1
set stretchY($w) 1
set Topmin($w) 1
set TopIsModule($w) 1
global is_guibuilder
wm geometry $w 138x50
wm geometry $w +160+176
wm resizable $w 1 1
wm minsize $w 138 50

set Name $w
set Parent $Name
}
proc mainwincr_floor_hci {nr} {
set w [win_name .floor_hci $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -borderwidth 2 -relief raised
set_onkill $w {destroy $w}
wm protocol $w WM_DELETE_WINDOW "destroy $w"
if {$nr > 0} {
wm title $w "floor_hci_${nr}"
} else {wm title $w "floor_hci"}
set Toppos($w) 1
set Topgeom($w) 1
set stretchX($w) 1
set stretchY($w) 1
set Topmin($w) 1
set TopIsModule($w) 1
global is_guibuilder
wm geometry $w 161x60
wm geometry $w +287+33
wm resizable $w 1 1
wm minsize $w 161 60

set Name $w
set Parent $Name
}
proc mainwincr_main1 {nr} {
set w [win_name .main1 $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -relief raised -background #0000ff
set_onkill $w {w_fc:stop_conference}
```

```
wm protocol $w WM_DELETE_WINDOW "w_fc:stop_conference"
if {$nr > 0} {
wm title $w "Tele-classroom Conference_${nr}"
} else {wm title $w "Tele-classroom Conference"}
set Toppos($w) 1
set Topgeom($w) 1
set stretchX($w) 1
set stretchY($w) 1
set Topmin($w) 1
set TopIsModule($w) 0
global is_guibuilder
wm geometry $w 307x394
wm geometry $w +256+308
wm resizable $w 1 1
wm minsize $w 307 394
if {!$is_guibuilder} {
  defaultpos_restore $w 1 0
  defaultpos_prepsave $w
}

set Name $w
set Parent $Name

set Name $Parent.l30
#----------------------------------------
label $Name -anchor w -relief sunken -text READY -textvariable
er_message
place $Name -x 5 -relx 0 -y -35 -rely 1 -width -68 -relwidth 1
-height 24 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.l31
#----------------------------------------
label $Name -anchor w -background blue -text Status
place $Name -x 84 -relx 0 -y -52 -rely 1 -width 44 -relwidth 0
-height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b35
#----------------------------------------
button $Name -command w_fc:stop_conference -highlightbackground
#0000ff -text Stop
place $Name -x -60 -relx 1 -y -40 -rely 1 -width 56 -relwidth 0
-height 32 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .main1.b35 <KeyPress-Return> {

  %W invoke

}

set Name $Parent.frame_requests
#----------------------------------------
frame $Name -borderwidth 2 -relief raised -background #64c8c8 -
height 50 -width 50
place $Name -x 6 -relx 0 -y -200 -rely 1 -width -11 -relwidth 1
-height 97 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.frame_requests.fb_listb2
#----------------------------------------
fancylistbox $Name -yscrollcommand "$Parent.frame_requests.sb92
set"
place $Name -x 6 -relx 0 -y 17 -rely 0 -width -34 -relwidth 1 -
height -23 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .main1.frame_requests.fb_listb2 <ButtonPress-1> {

  w_fc:show_listmenu %W %X %Y

}
bind .main1.frame_requests.fb_listb2 <KeyPress-Return> {

  w_fc:show_listmenu %W %X %Y

}
bind .main1.frame_requests.fb_listb2 <KeyPress-space> {

  w_fc:show_listmenu %W %X %Y

}

set Name $Parent.frame_requests.l48
#----------------------------------------
label $Name -anchor w -background #64c8c8 -text Requests
place $Name -x 5 -relx 0 -y 0 -rely 0 -width 64 -relwidth 0 -
height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.frame_requests.sb92
#----------------------------------------
scrollbar $Name -command "$Parent.frame_requests.fb_listb2
yview"
place $Name -x -29 -relx 1 -y 17 -rely 0 -width 23 -relwidth 0
-height -23 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.frame_class
#----------------------------------------
frame $Name -borderwidth 2 -relief raised -background #68c8c8 -
height 50 -width 50
place $Name -x 154 -relx 0 -y 23 -rely 0 -width -160 -relwidth
1 -height -229 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.frame_class.fb_listb1
#----------------------------------------
fancylistbox $Name -yscrollcommand "$Parent.frame_class.sb89
set"
place $Name -x 6 -relx 0 -y 17 -rely 0 -width -34 -relwidth 1 -
height -23 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .main1.frame_class.fb_listb1 <ButtonPress-1> {

  w_fc:show_listmenu %W %X %Y

}
bind .main1.frame_class.fb_listb1 <KeyPress-Return> {

  w_fc:show_listmenu %W %X %Y

}
bind .main1.frame_class.fb_listb1 <KeyPress-space> {

  w_fc:show_listmenu %W %X %Y

}

set Name $Parent.frame_class.l54
#----------------------------------------
label $Name -anchor w -background #64c8c8 -foreground #000000 -
text Participants
place $Name -x 5 -relx 0 -y 0 -rely 0 -width 75 -relwidth 0 -
height 15 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.frame_class.sb89
#----------------------------------------
scrollbar $Name -command "$Parent.frame_class.fb_listb1 yview"
place $Name -x -29 -relx 1 -y 17 -rely 0 -width 23 -relwidth 0
-height -23 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.floor
#----------------------------------------
frame $Name -borderwidth 2 -relief raised -background #64c8c8 -
height 50 -width 50
place $Name -x 6 -relx 0 -y 97 -rely 0 -width 143 -relwidth 0 -
height -303 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.floor.l79
#----------------------------------------
label $Name -anchor w -background #64c9c8 -text Access
place $Name -x 60 -relx 0 -y 35 -rely 0 -width 50 -relwidth 0 -
height 10 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.floor.b_withdraw
#----------------------------------------
button $Name -command w_fc:access:withdraw -highlightbackground
#64c8c8 -text Withdraw
place $Name -x 72 -relx 0 -y 4 -rely 0 -width 64 -relwidth 0 -
height 28 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .main1.floor.b_withdraw <KeyPress-Return> {

  %W invoke

}

set Name $Parent.floor.b_return
#----------------------------------------
button $Name -command w_fc:access:return -highlightbackground
#64c8c8 -text Return
place $Name -x 72 -relx 0 -y 52 -rely 0 -width 64 -relwidth 0 -
height 28 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .main1.floor.b_return <KeyPress-Return> {

  %W invoke

}

set Name $Parent.floor.b_floor_req
#----------------------------------------
button $Name -command w_freq_show -highlightbackground #64c8c8
-text Request
place $Name -x 4 -relx 0 -y 4 -rely 0 -width 64 -relwidth 0 -
height 28 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .main1.floor.b_floor_req <KeyPress-Return> {

  %W invoke

}

set Name $Parent.floor.l_image1
#----------------------------------------
button $Name -background #d9d3d3 -command w_fi:show -foreground
#000000 -highlightbackground #64c8c8 -highlightcolor #000000 -
image tr_no.gif -text Img
place $Name -x 11 -relx 0 -y 40 -rely 0 -width 40 -relwidth 0 -
height 40 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .main1.floor.l_image1 <KeyPress-Return> {

  %W invoke

}

set Name $Parent.b_panic
#----------------------------------------
button $Name -activeforeground #000000 -command
w_fc:grasp_floors -foreground #000000 -highlightbackground
#0000ff -highlightcolor #000000 -text PANIC
place $Name -x -116 -relx 1 -y -68 -rely 1 -width 48 -relwidth
0 -height 28 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .main1.b_panic <KeyPress-Return> {

  %W invoke

}
```

```
set Name $Parent.b121
#----------------------------------------
button $Name -command lm:show_win -highlightbackground #0000d9
-text Media
place $Name -x 6 -relx 0 -y -97 -rely 1 -width 51 -relwidth 0 -
height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .main1.b121 <KeyPress-Return> {

  %W invoke

}

set Name $Parent.b200
#----------------------------------------
button $Name -command w_txt:show_all -highlightbackground
#0000ff -text Text
place $Name -x 57 -relx 0 -y -97 -rely 1 -width 51 -relwidth 0
-height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .main1.b200 <KeyPress-Return> {

  %W invoke

}

set Name $Parent.b184
#----------------------------------------
button $Name -command wb:show -highlightbackground #0000ff -
text Whiteboard
place $Name -x 6 -relx 0 -y -69 -rely 1 -width 74 -relwidth 0 -
height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .main1.b184 <KeyPress-Return> {

  %W invoke

}

set Name $Parent.b185
#----------------------------------------
button $Name -command w_gmgr:show -highlightbackground #0000ff
-text M.Control
place $Name -x -68 -relx 1 -y -96 -rely 1 -width 64 -relwidth 0
-height 28 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .main1.b185 <KeyPress-Return> {

  %W invoke

}

set Name $Parent.f_info
#----------------------------------------
frame $Name -borderwidth 2 -relief raised -background #64c8c8 -
height 50 -width 50
place $Name -x 6 -relx 0 -y 23 -rely 0 -width 143 -relwidth 0 -
height 69 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_info.l193
#----------------------------------------
label $Name -anchor w -background #64c8c8 -highlightbackground
#0000ff -text Role
place $Name -x 0 -relx 0 -y 20 -rely 0 -width 30 -relwidth 0 -
height 25 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_info.l36
#----------------------------------------
label $Name -anchor w -background #64c8c8 -text Name
place $Name -x 0 -relx 0 -y 44 -rely 0 -width -100 -relwidth 1
-height -48 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_info.l65
#----------------------------------------
label $Name -anchor w -relief sunken -text TEACHER -
textvariable w_fc_actor_indication
place $Name -x 40 -relx 0 -y 23 -rely 0 -width 97 -relwidth 0 -
height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_info.l37
#----------------------------------------
label $Name -anchor w -relief sunken -textvariable
w_fc_name_out
place $Name -x 40 -relx 0 -y 46 -rely 0 -width 97 -relwidth 0 -
height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_info.l38
#----------------------------------------
label $Name -anchor w -background #64c8c8 -borderwidth 1 -
relief sunken -textvariable w_fc_conf_id
place $Name -x 6 -relx 0 -y 6 -rely 0 -width 131 -relwidth 0 -
height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.fmenu
#----------------------------------------
frame $Name -borderwidth 2 -height 50 -width 50
place $Name -x 0 -relx 0 -y 0 -rely 0 -width 0 -relwidth 1 -
height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.fmenu.m_fc
#----------------------------------------
menubutton $Name -text FloorControl -underline 5
set Menu_string($Name) {{
  {Panic command "w_fc:grasp_floors" -underline 0}
  {"Enable speech" menu {
  } -tearoff 0}
  {"Disable speech" menu {
  } -tearoff 0}
  {"Refuse speech" menu {
  } -tearoff 0}
} -tearoff 0
}
$Name configure -menu $Name.m
eval "make_menu $Name $Menu_string($Name)"
place $Name -x 112 -relx 0 -y -1 -rely 0 -width 80 -relwidth 0
-height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.fmenu.m_w
#----------------------------------------
menubutton $Name -text Window -underline 2
set Menu_string($Name) {{
{Media command lm:show_win -underline 0}
{Text menu {
  {All command w_txt:show_all}
} -tearoff 0}
{Whiteboard command {w_whiteboard:show} -underline 0}
{separator}
{{Meeting Control} command {wshow .groupmgr} -underline 8}
} -tearoff 0
}
$Name configure -menu $Name.m
eval "make_menu $Name $Menu_string($Name)"
place $Name -x 192 -relx 0 -y -1 -rely 0 -width 40 -relwidth 0
-height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.fmenu.m_f
#----------------------------------------
menubutton $Name -text File -underline 0
set Menu_string($Name) {{
{{Save window-positions} command {main:save_winpos}}
{separator}
{Quit command w_fc:stop_conference -underline 0}
} -tearoff 0
}
$Name configure -menu $Name.m
eval "make_menu $Name $Menu_string($Name)"
place $Name -x 0 -relx 0 -y -2 -rely 0 -width 46 -relwidth 0 -
height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.fmenu.m_fa
#----------------------------------------
menubutton $Name -text FloorAccess -underline 5
set Menu_string($Name) {{
  {Request command "w_freq_show" -underline 0}
  {Withdraw command "w_fc:access:withdraw" -underline 0}
  {Return command "w_fc:access:return" -underline 3}
  {separator}
  {Show command "w_fi:show" -underline 0}
} -tearoff 0
```

```
}
$Name configure -menu $Name.m
eval "make_menu $Name $Menu_string($Name)"
place $Name -x 40 -relx 0 -y -1 -rely 0 -width 72 -relwidth 0 -
height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.fmenu.m203
#----------------------------------------
menubutton $Name -borderwidth 0 -text Help -underline 0
set Menu_string($Name) {{
  {{Not yet Implemented} command {}}
} -tearoff 0



}
$Name configure -menu $Name.m
eval "make_menu $Name $Menu_string($Name)"
place $Name -x -28 -relx 1 -y -3 -rely 0 -width 24 -relwidth 0
-height 20 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b201
#----------------------------------------
button $Name -command w_fc:show_faces -highlightbackground
#0000ff -text Faces
place $Name -x -116 -relx 1 -y -96 -rely 1 -width 48 -relwidth
0 -height 28 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
}
proc mainwincr_media1 {nr} {
set w [win_name .media1 $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -relief sunken -background #0000ff
set_onkill $w {lm:hide_win}
wm protocol $w WM_DELETE_WINDOW "lm:hide_win"
if {$nr > 0} {
wm title $w "Local Media Control_${nr}"
} else {wm title $w "Local Media Control"}
set Toppos($w) 1
set Topgeom($w) 0
set stretchX($w) 0
set stretchY($w) 0
set Topmin($w) 1
set TopIsModule($w) 0
global is_guibuilder
wm geometry $w +388+162
wm resizable $w 0 0
wm minsize $w 205 397
if {!$is_guibuilder} {
  defaultpos_restore $w 1 0
  defaultpos_prepsave $w
}

set Name $w
set Parent $Name
bind .media1 <Alt-h> {

  lm:hide_win

}

set Name $Parent.f_vid_in
#----------------------------------------
frame $Name -borderwidth 2 -relief raised -background #64c8c8 -
height 52 -highlightbackground #64c8c8 -width 196
pack $Name -anchor nw -expand 0 -fill x -ipadx 0 -ipady 0 -padx
0 -pady 0 -side top
placement_store "$Name" "pack"

set Name $Parent.f_vid_in.l142
#----------------------------------------
label $Name -anchor w -background #64c8c8 -borderwidth 1 -
highlightbackground #64c8c8 -relief raised -text {Video In}
place $Name -x 0 -relx 0 -y 0 -rely 0 -width 65 -relwidth 0 -
height 15 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .media1.f_vid_in.l142 <ButtonPress-1> {

  lm:vi:toggle_frame

}

set Name $Parent.f_vid_in.b143
#----------------------------------------
checkbutton $Name -anchor w -background #64c8c8 -command
lm:vi:enable -highlightbackground #64c8c8 -text Mute -variable
lm_vi_bmute -width 8
place $Name -x 74 -relx 0 -y 0 -rely 0 -width 63 -relwidth 0 -
height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_vid_in.l_msg
#----------------------------------------
label $Name -background #64c8c8 -borderwidth 1 -
highlightbackground #64c8c8 -relief sunken -textvariable
lm_vi_msg
place $Name -x 6 -relx 0 -y 23 -rely 0 -width 183 -relwidth 0 -
height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_vid_out
#----------------------------------------
frame $Name -borderwidth 3 -relief groove -background #64c8c8 -
height 76 -highlightbackground #64c8c8 -width 196
pack $Name -anchor center -expand 0 -fill x -ipadx 0 -ipady 0 -
padx 0 -pady 0 -side top
placement_store "$Name" "pack"

set Name $Parent.f_vid_out.l163
#----------------------------------------
label $Name -anchor w -background #64c8c8 -borderwidth 1 -
highlightbackground #64c8c8 -relief raised -text {Video Out}
place $Name -x 0 -relx 0 -y 0 -rely 0 -width 65 -relwidth 0 -
height 15 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .media1.f_vid_out.l163 <ButtonPress-1> {

  lm:vo:toggle_frame

}

set Name $Parent.f_vid_out.s164
#----------------------------------------
scale $Name -background #64c8c8 -command lm:vo:set_vsize -
highlightbackground #64c8c8 -orient horizontal -showvalue 0 -to
4.0 -variable lm_vo_size
place $Name -x 34 -relx 0 -y 23 -rely 0 -width 74 -relwidth 0 -
height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_vid_out.l165
#----------------------------------------
label $Name -anchor w -background #64c8c8 -highlightbackground
#64c8c8 -text Size
place $Name -x 5 -relx 0 -y 25 -rely 0 -width 30 -relwidth 0 -
height 15 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_vid_out.b166
#----------------------------------------
checkbutton $Name -anchor w -background #64c8c8 -command
lm:vo:change -highlightbackground #64c8c8 -text Mute -variable
lm_vo_bmute -width 8
place $Name -x 70 -relx 0 -y 0 -rely 0 -width 65 -relwidth 0 -
height 15 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_vid_out.b167
#----------------------------------------
checkbutton $Name -anchor w -background #64c8c8 -command
lm:vo:change -highlightbackground #64c8c8 -text preview -
variable lm_vo_preview -width 8
place $Name -x 109 -relx 0 -y 23 -rely 0 -width 74 -relwidth 0
-height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_vid_out.lmsg
#----------------------------------------
label $Name -background #64c8c8 -borderwidth 1 -
highlightbackground #64c8c8 -relief sunken -textvariable
lm_vo_msg
place $Name -x 6 -relx 0 -y 46 -rely 0 -width 177 -relwidth 0 -
height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_audio_in2
#----------------------------------------
frame $Name -borderwidth 3 -relief groove -background #64c8c8 -
height 100 -highlightbackground #64c8c8 -width 116
pack $Name -anchor nw -expand 0 -fill x -ipadx 0 -ipady 0 -padx
0 -pady 0 -side top
placement_store "$Name" "pack"

set Name $Parent.f_audio_in2.l172
#----------------------------------------
label $Name -anchor w -background #64c8c8 -borderwidth 1 -
highlightbackground #64c8c8 -relief raised -text {Audio In}
place $Name -x 0 -relx 0 -y 0 -rely 0 -width 65 -relwidth 0 -
height 15 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .media1.f_audio_in2.l172 <ButtonPress-1> {

  lm:ai:toggle_frame
}

set Name $Parent.f_audio_in2.s173
#----------------------------------------
scale $Name -background #64c8c8 -command sy:set_volume_out -
highlightbackground #64c8c8 -orient horizontal -showvalue 0 -to
255.0 -variable lm_ai_volume
place $Name -x 72 -relx 0 -y 44 -rely 0 -width 112 -relwidth 0
-height 24 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_audio_in2.l174
#----------------------------------------
label $Name -anchor w -background #64c8c8 -highlightbackground
#64c8c8 -text Volume
place $Name -x 15 -relx 0 -y 50 -rely 0 -width 45 -relwidth 0 -
height 15 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_audio_in2.b178
#----------------------------------------
checkbutton $Name -anchor w -background #64c8c8 -command
lm:ai:enable -highlightbackground #64c8c8 -text Mute -variable
lm_ai_bmute -width 8
place $Name -x 70 -relx 0 -y 0 -rely 0 -width 65 -relwidth 0 -
height 15 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_audio_in2.b137
#----------------------------------------
```

```
checkbutton $Name -anchor w -background #64c8c8 -command
lm:ai:set_source -highlightbackground #64c8c8 -text Speaker -
variable lm_ai_speaker -width 8
place $Name -x 0 -relx 0 -y 23 -rely 0 -width 74 -relwidth 0 -
height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_audio_in2.b139
#----------------------------------------
checkbutton $Name -anchor w -background #64c8c8 -command
lm:ai:set_source -highlightbackground #64c8c8 -text Head -
variable lm_ai_head -width 8
place $Name -x 74 -relx 0 -y 23 -rely 0 -width 57 -relwidth 0 -
height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_audio_in2.b141
#----------------------------------------
checkbutton $Name -anchor w -background #64c8c8 -command
lm:ai:set_source -highlightbackground #64c8c8 -text Aux -
variable lm_ai_aux -width 8
place $Name -x 131 -relx 0 -y 23 -rely 0 -width 46 -relwidth 0
-height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_audio_in2.l_msg
#----------------------------------------
label $Name -background #64c8c8 -borderwidth 1 -
highlightbackground #64c8c8 -relief sunken -textvariable
lm_ai_msg
place $Name -x 6 -relx 0 -y 74 -rely 0 -width 177 -relwidth 0 -
height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_audio_out
#----------------------------------------
frame $Name -borderwidth 2 -relief raised -background #64c8c8 -
height 124 -highlightbackground #64c8c8 -width 188
pack $Name -anchor n -expand 0 -fill x -ipadx 0 -ipady 0 -padx
0 -pady 0 -side top
placement_store "$Name" "pack"

set Name $Parent.f_audio_out.b181
#----------------------------------------
radiobutton $Name -anchor w -background #64c8c8 -command
lm:ao:set_source -highlightbackground #64c8c8 -text Microphone
-value 1 -variable lm_ao_source -width 8
place $Name -x 11 -relx 0 -y 23 -rely 0 -width 103 -relwidth 0
-height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_audio_out.b182
#----------------------------------------
radiobutton $Name -anchor w -background #64c8c8 -command
lm:ao:set_source -highlightbackground #64c8c8 -text Aux -value
2 -variable lm_ao_source -width 8
place $Name -x 114 -relx 0 -y 23 -rely 0 -width 57 -relwidth 0
-height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_audio_out.l184
#----------------------------------------
label $Name -anchor w -background #64c8c8 -highlightbackground
#64c8c8 -text {Rec. Vol.}
place $Name -x 5 -relx 0 -y 40 -rely 0 -width 55 -relwidth 0 -
height 30 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_audio_out.s185
#----------------------------------------
scale $Name -background #64c8c8 -command sy:set_volume_in -
highlightbackground #64c8c8 -orient horizontal -showvalue 0 -to
255.0 -variable lm_ao_volume_src
place $Name -x 69 -relx 0 -y 46 -rely 0 -width 112 -relwidth 0
-height 24 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_audio_out.l188
#----------------------------------------
label $Name -anchor w -background #64c8c8 -borderwidth 1 -
highlightbackground #64c8c8 -relief raised -text {Audio Out}
place $Name -x 0 -relx 0 -y 0 -rely 0 -width 65 -relwidth 0 -
height 15 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .media1.f_audio_out.l188 <ButtonPress-1> {

  lm:ao:toggle_frame
}

set Name $Parent.f_audio_out.s206
#----------------------------------------
scale $Name -background #64c8c8 -command sy:set_volume_mon -
highlightbackground #64c8c8 -orient horizontal -showvalue 0 -to
255.0 -variable lm_ao_volume_mon
place $Name -x 69 -relx 0 -y 74 -rely 0 -width 112 -relwidth 0
-height 24 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_audio_out.l207
#----------------------------------------
label $Name -anchor w -background #64c8c8 -highlightbackground
#64c8c8 -text {Mon. Vol.}
place $Name -x 5 -relx 0 -y 75 -rely 0 -width 55 -relwidth 0 -
height 15 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_audio_out.l_msg
#----------------------------------------
label $Name -background #64c8c8 -borderwidth 1 -
highlightbackground #64c8c8 -relief sunken -textvariable
lm_ao_msg
place $Name -x 6 -relx 0 -y 103 -rely 0 -width 171 -relwidth 0
-height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_audio_out.b_speak
#----------------------------------------
button $Name -highlightbackground #64c8c8 -text Speak
place $Name -x 76 -relx 0 -y 0 -rely 0 -width 60 -relwidth 0 -
height 20 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .media1.f_audio_out.b_speak <ButtonPress-1> {

  global audio_out_button_mute

  set audio_out_button_mute 0
  lm:ao:enable

}
bind .media1.f_audio_out.b_speak <ButtonRelease-1> {

  global audio_out_button_mute

  set audio_out_button_mute 1
  lm:ao:enable

}

set Name $Parent.f_audio_out.b_mute
#----------------------------------------
checkbutton $Name -anchor w -background #64c8c8 -command
lm:ao:enable -foreground #000000 -highlightbackground #64c8c8 -
text Mute -variable lm_ao_bmute -width 8
place $Name -x 70 -relx 0 -y 0 -rely 0 -width 65 -relwidth 0 -
height 15 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_con
#----------------------------------------
frame $Name -borderwidth 2 -background #0000ff -height 40 -
highlightbackground #64c8c8 -width 205
pack $Name -anchor n -expand 0 -fill x -ipadx 0 -ipady 0 -padx
0 -pady 0 -side top
placement_store "$Name" "pack"

set Name $Parent.f_con.b190
#----------------------------------------
button $Name -command lm:hide_win -text Hide -underline 0
place $Name -x 40 -relx 0 -y 6 -rely 0 -width 103 -relwidth 0 -
height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .media1.f_con.b190 <KeyPress-Return> {

  %W invoke

}

set Name $Parent.dummy
#----------------------------------------
frame $Name -borderwidth 2 -background #0000ff -height 5 -width
182
pack $Name -anchor center -expand 0 -fill none -ipadx 0 -ipady
0 -padx 0 -pady 0 -side top
placement_store "$Name" "pack"
}
proc mainwincr_supercontrol {nr} {
set w [win_name .supercontrol $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w
set_onkill $w {#}
wm protocol $w WM_DELETE_WINDOW "#"
if {$nr > 0} {
wm title $w "supercontrol Window_${nr}"
} else {wm title $w "supercontrol Window"}
set Toppos($w) 1
set Topgeom($w) 1
set stretchX($w) 1
set stretchY($w) 1
set Topmin($w) 1
set TopIsModule($w) 0
global is_guibuilder
wm geometry $w 412x719
wm geometry $w +860+23
wm resizable $w 1 1
wm minsize $w 412 719
if {!$is_guibuilder} {
  defaultpos_restore $w 1 0
  defaultpos_prepsave $w
}

set Name $w
set Parent $Name

set Name $Parent.l30
#----------------------------------------
label $Name -anchor w -relief sunken -text READY -textvariable
er_message
place $Name -x 6 -relx 0 -y -29 -rely 1 -width -74 -relwidth 1
-height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.e27
#----------------------------------------
entry $Name -background white -textvariable sc_pcommand
place $Name -x 6 -relx 0 -y 17 -rely 0 -width -11 -relwidth 1 -
height 23 -relheight 0 -anchor nw -bordermode inside
```

```
placement_store "$Name" "place"
bind .supercontrol.e27 <KeyPress-Return> {

  w_sc:pipe_command

}

set Name $Parent.l29
#----------------------------------------
label $Name -anchor w -text {Network-layer Command}
place $Name -x 6 -relx 0 -y 0 -rely 0 -width 223 -relwidth 0 -
height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b45
#----------------------------------------
button $Name -command w_sc:clearlists -text Clear
place $Name -x -63 -relx 1 -y -34 -rely 1 -width 57 -relwidth 0
-height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_out
#----------------------------------------
frame $Name -borderwidth 2 -relief raised -height 50 -width 50
place $Name -x 0 -relx 0.01429 -y 0 -rely 0.1286 -width 0 -
relwidth 0.9714 -height 0 -relheight 0.2714 -anchor nw -
bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_out.l139
#----------------------------------------
label $Name -anchor w -text transmitted
place $Name -x 5 -relx 0 -y 0 -rely 0 -width 85 -relwidth 0 -
height 15 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_out.sb140
#----------------------------------------
scrollbar $Name -command "$Parent.f_out.lb yview"
place $Name -x -29 -relx 1 -y 17 -rely 0 -width 23 -relwidth 0
-height -23 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_out.lb
#----------------------------------------
listbox $Name -yscrollcommand "$Parent.f_out.sb140 set"
place $Name -x 6 -relx 0 -y 17 -rely 0 -width -34 -relwidth 1 -
height -23 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_in
#----------------------------------------
frame $Name -borderwidth 2 -relief raised -height 50 -width 50
place $Name -x 0 -relx 0.01429 -y 0 -rely 0.4 -width 0 -
relwidth 0.9714 -height 0 -relheight 0.2857 -anchor nw -
bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_in.l144
#----------------------------------------
label $Name -anchor w -text Received
place $Name -x 5 -relx 0 -y 0 -rely 0 -width 51 -relwidth 0 -
height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_in.sb146
#----------------------------------------
scrollbar $Name -command "$Parent.f_in.lb yview"
place $Name -x -29 -relx 1 -y 17 -rely 0 -width 23 -relwidth 0
-height -23 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_in.lb
#----------------------------------------
listbox $Name -yscrollcommand "$Parent.f_in.sb146 set"
place $Name -x 6 -relx 0 -y 17 -rely 0 -width -34 -relwidth 1 -
height -23 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_debug
#----------------------------------------
frame $Name -borderwidth 2 -relief raised -height 50 -width 50
place $Name -x 0 -relx 0.01429 -y 0 -rely 0.6857 -width 0 -
relwidth 0.9714 -height 0 -relheight 0.2571 -anchor nw -
bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_debug.l128
#----------------------------------------
label $Name -anchor w -text Debug
place $Name -x 5 -relx 0 -y 0 -rely 0 -width 51 -relwidth 0 -
height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_debug.sb129
#----------------------------------------
scrollbar $Name -command "$Parent.f_debug.lb yview"
place $Name -x -29 -relx 1 -y 17 -rely 0 -width 23 -relwidth 0
-height -23 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_debug.lb
#----------------------------------------
listbox $Name -yscrollcommand "$Parent.f_debug.sb129 set"
place $Name -x 6 -relx 0 -y 17 -rely 0 -width -34 -relwidth 1 -
height -23 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.e206
#----------------------------------------
entry $Name -background white -textvariable sc_tcommand
place $Name -x 6 -relx 0 -y 63 -rely 0 -width -11 -relwidth 1 -
height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .supercontrol.e206 <KeyRelease-Return> {

  w_sc:tcl_command

}

set Name $Parent.l208
#----------------------------------------
label $Name -anchor w -text {Tcl/Tk Command}
place $Name -x 6 -relx 0 -y 46 -rely 0 -width 125 -relwidth 0 -
height 15 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
}
proc mainwincr_text1 {nr} {
set w [win_name .text1 $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -background #64c8c8
set_onkill $w {whide $w}
wm protocol $w WM_DELETE_WINDOW "whide $w"
if {$nr > 0} {
wm title $w "Text Board_${nr}"
} else {wm title $w "Text Board"}
set Toppos($w) 0
set Topgeom($w) 1
set stretchX($w) 1
set stretchY($w) 1
set Topmin($w) 1
set TopIsModule($w) 0
global is_guibuilder
wm geometry $w 381x202
wm resizable $w 1 1
wm minsize $w 381 202
if {!$is_guibuilder} {
  defaultpos_restore $w 1 0
  defaultpos_prepsave $w
}

set Name $w
set Parent $Name

set Name $Parent.sb96
#----------------------------------------
scrollbar $Name -command "$Parent.fb_textout yview" -
highlightbackground #64c8c8
place $Name -x -34 -relx 1 -y 74 -rely 0 -width 23 -relwidth 0
-height -114 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.l_out_msg
#----------------------------------------
label $Name -anchor w -background #64c8c8 -borderwidth 1 -
relief sunken
place $Name -x 74 -relx 0 -y 40 -rely 0 -width -143 -relwidth 1
-height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.entry
#----------------------------------------
entry $Name -background #ffffff -highlightbackground #64c8c8 -
textvariable in_text_entry
place $Name -x 6 -relx 0 -y 6 -rely 0 -width -74 -relwidth 1 -
height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.in_message
#----------------------------------------
label $Name -anchor w -background #64c8c8 -borderwidth 1 -
relief sunken
place $Name -x 69 -relx 0 -y -34 -rely 1 -width -143 -relwidth
1 -height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.fb_textout
#----------------------------------------
fancylistbox $Name -yscrollcommand "$Parent.sb96 set"
place $Name -x 6 -relx 0 -y 74 -rely 0 -width -40 -relwidth 1 -
height -114 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b_hide
#----------------------------------------
button $Name -highlightbackground #64c8c8 -text Hide -underline
0
place $Name -x -63 -relx 1 -y -34 -rely 1 -width 57 -relwidth 0
-height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b_c_entry
#----------------------------------------
button $Name -highlightbackground #64c8c8 -text Clear -
underline 2
place $Name -x 6 -relx 0 -y 40 -rely 0 -width 46 -relwidth 0 -
height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b_send
#----------------------------------------
button $Name -highlightbackground #64c8c8 -text Send -underline
0
place $Name -x -63 -relx 1 -y 6 -rely 0 -width 57 -relwidth 0 -
height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
```

```
set Name $Parent.b_c_text
#----------------------------------------
button $Name -highlightbackground #64c8c8 -text Clear -
underline 0
place $Name -x 6 -relx 0 -y -34 -rely 1 -width 46 -relwidth 0 -
height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
}
proc mainwincr_freq_popup {nr} {
set w [win_name .freq_popup $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -borderwidth 2 -relief raised -background #64c8c8
set_onkill $w {w_freq_hide}
wm protocol $w WM_DELETE_WINDOW "w_freq_hide"
if {$nr > 0} {
wm title $w "FLoor Request_${nr}"
} else {wm title $w "FLoor Request"}
set Toppos($w) 1
set Topgeom($w) 1
set stretchX($w) 0
set stretchY($w) 0
set Topmin($w) 1
set TopIsModule($w) 0
global is_guibuilder
wm geometry $w 236x141
wm geometry $w +264+165
wm resizable $w 0 0
wm minsize $w 236 141
if {!$is_guibuilder} {
  defaultpos_restore $w 1 0
  defaultpos_prepsave $w
}

set Name $w
set Parent $Name
bind .freq_popup <Alt-o> {

  w_freq_apply

}
bind .freq_popup <Alt-c> {

  w_freq_hide

}
bind .freq_popup <Alt-f> {

  w_freq_showfmenu

}
bind .freq_popup <Alt-r> {

  focus .freq_popup.entry1

}
bind .freq_popup <KeyPress-Escape> {

   w_freq_hide

}

set Name $Parent.b91
#----------------------------------------
button $Name -command w_freq_hide -highlightbackground #64c8c8
-text Cancel -underline 0
place $Name -x 17 -relx 0 -y 97 -rely 0 -width 69 -relwidth 0 -
height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .freq_popup.b91 <KeyPress-Return> {

  %W invoke

}

set Name $Parent.l103
#----------------------------------------
label $Name -anchor w -background #64c8c8 -text Floors -
underline 0
place $Name -x 10 -relx 0 -y 15 -rely 0 -width 40 -relwidth 0 -
height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.l106
#----------------------------------------
label $Name -anchor w -background #64c8c8 -text Reason -
underline 0
place $Name -x 10 -relx 0 -y 45 -rely 0 -width 50 -relwidth 0 -
height 15 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b110
#----------------------------------------
button $Name -command w_freq_apply -highlightbackground #64c8c8
-text OK -underline 0
place $Name -x -63 -relx 1 -y 97 -rely 0 -width 51 -relwidth 0
-height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .freq_popup.b110 <KeyPress-Return> {

  %W invoke

}

set Name $Parent.entry1
#----------------------------------------
entry $Name -background white -highlightbackground #64c8c8 -
textvariable w_freq_reason
place $Name -x 11 -relx 0 -y 63 -rely 0 -width -23 -relwidth 1
-height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .freq_popup.entry1 <KeyPress-Return> {

  w_freq_apply

}

set Name $Parent.m_floormenu
#----------------------------------------
menubutton $Name -anchor w -relief raised -text {Audio 1 + Text
1} -textvariable w_freq_choiceview
set Menu_string($Name) {{
} -background white -activebackground black -activeforeground
white

}
$Name configure -menu $Name.m
eval "make_menu $Name $Menu_string($Name)"
place $Name -x 63 -relx 0 -y 17 -rely 0 -width -69 -relwidth 1
-height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .freq_popup.m_floormenu <KeyPress-Return> {

  %W invoke

}
}
proc mainwincr_fref_popup {nr} {
set w [win_name .fref_popup $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -background #64c8c8 -highlightbackground #64c8c8
set_onkill $w {w_fp:hide}
wm protocol $w WM_DELETE_WINDOW "w_fp:hide"
if {$nr > 0} {
wm title $w "Refuse Request_${nr}"
} else {wm title $w "Refuse Request"}
set Toppos($w) 1
set Topgeom($w) 1
set stretchX($w) 0
set stretchY($w) 0
set Topmin($w) 1
set TopIsModule($w) 0
global is_guibuilder
wm geometry $w 249x107
wm geometry $w +215+246
wm resizable $w 0 0
wm minsize $w 249 107
if {!$is_guibuilder} {
  defaultpos_restore $w 1 0
  defaultpos_prepsave $w
}

set Name $w
set Parent $Name
bind .fref_popup <Alt-c> {

  w_fp:hide

}
bind .fref_popup <Alt-o> {

  w_fp:ok

}
bind .fref_popup <KeyPress-Escape> {

  w_fp:hide

}

set Name $Parent.l102
#----------------------------------------
label $Name -anchor w -background #64c8c8 -text Reason
```

```
place $Name -x 10 -relx 0 -y 5 -rely 0 -width 55 -relwidth 0 -
height 15 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b103
#----------------------------------------
button $Name -command w_fp:hide -highlightbackground #64c8c8 -
text Cancel -underline 0
place $Name -x 11 -relx 0 -y 63 -rely 0 -width 51 -relwidth 0 -
height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .fref_popup.b103 <KeyPress-Return> {

  %W invoke

}

set Name $Parent.b104
#----------------------------------------
button $Name -command w_fp:ok -highlightbackground #64c8c8 -
text OK -underline 0
place $Name -x -63 -relx 1 -y 63 -rely 0 -width 57 -relwidth 0
-height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .fref_popup.b104 <KeyPress-Return> {

  %W invoke

}

set Name $Parent.entry1
#----------------------------------------
entry $Name -background white -highlightbackground #64c8c8 -
textvariable w_fp_reason
place $Name -x 11 -relx 0 -y 23 -rely 0 -width -17 -relwidth 1
-height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .fref_popup.entry1 <KeyPress-Return> {

  w_fp:ok

}
}
proc mainwincr_groupmgr {nr} {
set w [win_name .groupmgr $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -background #64c8c8
set_onkill $w {whide $w}
wm protocol $w WM_DELETE_WINDOW "whide $w"
if {$nr > 0} {
wm title $w "Group Management_${nr}"
} else {wm title $w "Group Management"}
set Toppos($w) 1
set Topgeom($w) 1
set stretchX($w) 1
set stretchY($w) 1
set Topmin($w) 1
set TopIsModule($w) 0
global is_guibuilder
wm geometry $w 331x403
wm geometry $w +32+542
wm resizable $w 1 1
wm minsize $w 331 403
if {!$is_guibuilder} {
  defaultpos_restore $w 1 0
  defaultpos_prepsave $w
}

set Name $w
set Parent $Name
bind .groupmgr <Alt-h> {

 .groupmgr.b_hide invoke

}

set Name $Parent.f128
#----------------------------------------
frame $Name -borderwidth 2 -relief raised -background #0000ff -
height 50 -width 50
place $Name -x 194 -relx 0 -y 6 -rely 0 -width -200 -relwidth 1
-height -149 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f128.fb134
#----------------------------------------
fancylistbox $Name -yscrollcommand "$Parent.f128.sb137 set"
place $Name -x 6 -relx 0 -y 17 -rely 0 -width -34 -relwidth 1 -
height -23 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f128.l135
#----------------------------------------
label $Name -anchor w -background #0000ff -foreground #000000 -
text Participants
place $Name -x 5 -relx 0 -y 0 -rely 0 -width 75 -relwidth 0 -
height 15 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f128.sb137
#----------------------------------------
scrollbar $Name -command "$Parent.f128.fb134 yview"
place $Name -x -29 -relx 1 -y 17 -rely 0 -width 23 -relwidth 0
-height -23 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f138
#----------------------------------------
frame $Name -borderwidth 2 -relief raised -background #0000ff -
height 50 -width 50
place $Name -x 5 -relx 0 -y -140 -rely 1 -width -11 -relwidth 1
-height 91 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f138.fb180
#----------------------------------------
fancylistbox $Name -yscrollcommand "$Parent.f138.sb183 set"
place $Name -x 6 -relx 0 -y 23 -rely 0 -width -34 -relwidth 1 -
height -29 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f138.l181
#----------------------------------------
label $Name -anchor w -background #0000ff -text {Control
Requests}
place $Name -x 5 -relx 0 -y 5 -rely 0 -width 110 -relwidth 0 -
height 10 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f138.sb183
#----------------------------------------
scrollbar $Name -command "$Parent.f138.fb180 yview"
place $Name -x -29 -relx 1 -y 23 -rely 0 -width 23 -relwidth 0
-height -29 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.l155
#----------------------------------------
label $Name -relief sunken -textvariable my_name
place $Name -x 74 -relx 0 -y 17 -rely 0 -width 114 -relwidth 0
-height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.l157
#----------------------------------------
label $Name -relief sunken -textvariable refl_adr
place $Name -x 74 -relx 0 -y 63 -rely 0 -width 109 -relwidth 0
-height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.l160
#----------------------------------------
label $Name -anchor w -background #64c8c8 -text Name
place $Name -x 6 -relx 0 -y 17 -rely 0 -width 29 -relwidth 0 -
height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.l162
#----------------------------------------
label $Name -anchor w -background #64c8c8 -text Role
place $Name -x 6 -relx 0 -y 63 -rely 0 -width 63 -relwidth 0 -
height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f157
#----------------------------------------
frame $Name -borderwidth 2 -relief raised -background #0000ff -
height 50 -width 50
place $Name -x 6 -relx 0 -y -269 -rely 1 -width 189 -relwidth 0
-height 126 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f157.l158
#----------------------------------------
label $Name -anchor w -background #0000ff -text {Floor Control}
place $Name -x 0 -relx 0.025 -y 0 -rely 0.0125 -width 0 -
relwidth 0.4375 -height 0 -relheight 0.125 -anchor nw -
bordermode inside
placement_store "$Name" "place"

set Name $Parent.f157.m159
#----------------------------------------
menubutton $Name -relief raised -text Request
set Menu_string($Name) {{
  {Audio+Text command "w_gmgr:req_floor_au_txt"}
  {Audio command "w_gmgr:req_floor_audio"}
  {Text command "w_gmgr:req_floor_text"}
}
}
$Name configure -menu $Name.m
eval "make_menu $Name $Menu_string($Name)"
place $Name -x 40 -relx 0 -y -133 -rely 0 -width -128 -relwidth
1 -height -96 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f157.b160
#----------------------------------------
button $Name -text Withdraw
place $Name -x 103 -relx 0 -y 23 -rely 0 -width 69 -relwidth 0
-height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f157.b161
#----------------------------------------
button $Name -text Return
place $Name -x 103 -relx 0 -y 63 -rely 0 -width 69 -relwidth 0
-height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
```

```
set Name $Parent.f157.b162
#----------------------------------------
button $Name -text Request
place $Name -x 11 -relx 0 -y 23 -rely 0 -width 69 -relwidth 0 -
height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f157.l163
#----------------------------------------
label $Name -background #0000ff -text image
place $Name -x 29 -relx 0 -y 57 -rely 0 -width 51 -relwidth 0 -
height 51 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b165
#----------------------------------------
button $Name -highlightcolor #000000 -text {Floor Defenition}
place $Name -x 6 -relx 0 -y -40 -rely 1 -width 103 -relwidth 0
-height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b_hide
#----------------------------------------
button $Name -command w_gmgr:hide -text Hide -underline 0
place $Name -x 0 -relx 0.7714 -y -40 -rely 1 -width 0 -relwidth
0.2 -height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
}
proc mainwincr_trans_start {nr} {
set w [win_name .trans_start $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -relief raised -background #64c8c8
set_onkill $w {w_st1:quit}
wm protocol $w WM_DELETE_WINDOW "w_st1:quit"
if {$nr > 0} {
wm title $w "Start DTC in testmode_${nr}"
} else {wm title $w "Start DTC in testmode"}
set Toppos($w) 1
set Topgeom($w) 1
set stretchX($w) 0
set stretchY($w) 0
set Topmin($w) 1
set TopIsModule($w) 0
global is_guibuilder
wm geometry $w 249x317
wm geometry $w +410+284
wm resizable $w 0 0
wm minsize $w 249 317
if {!$is_guibuilder} {
  defaultpos_restore $w 1 0
  defaultpos_prepsave $w
}

set Name $w
set Parent $Name
bind .trans_start <Alt-s> {

  w_st1:start_conference

}
bind .trans_start <Alt-q> {

  w_st1:quit

}
bind .trans_start <Key-Return> {}

set Name $Parent.l_message
#----------------------------------------
label $Name -anchor w -background #64c8c8 -relief sunken -text
READY -textvariable er_message
place $Name -x 12 -relx 0 -y 280 -rely 0 -width 160 -relwidth 0
-height 24 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.l6
#----------------------------------------
label $Name -anchor w -background #64c8c8 -text Status
place $Name -x 8 -relx 0 -y 264 -rely 0 -width 63 -relwidth 0 -
height 11 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.e_refaddr
#----------------------------------------
entry $Name -background white -highlightbackground #64c8c8 -
textvariable wst1_refl_address
place $Name -x 6 -relx 0 -y 69 -rely 0 -width 109 -relwidth 0 -
height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .trans_start.e_refaddr <KeyPress> {

.trans_start.l_message configure -text " "

}

set Name $Parent.l11
#----------------------------------------
label $Name -anchor w -background #64c8c8 -text {Reflector
Address}
place $Name -x 5 -relx 0 -y 50 -rely 0 -width 110 -relwidth 0 -
height 15 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b_go
#----------------------------------------
button $Name -command w_st1:start_conference -height 1 -
highlightbackground #64c8c8 -text Start -underline 0 -width 6
place $Name -x 24 -relx 0 -y 196 -rely 0 -width 63 -relwidth 0
-height 34 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b_quit
#----------------------------------------
button $Name -command w_st1:quit -highlightbackground #64c8c8 -
text Quit -underline 0
place $Name -x 180 -relx 0 -y 276 -rely 0 -width 60 -relwidth 0
-height 33 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.l13
#----------------------------------------
label $Name -anchor w -background #64c8c8 -text Name
place $Name -x 5 -relx 0 -y 5 -rely 0 -width 40 -relwidth 0 -
height 10 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f0
#----------------------------------------
frame $Name -borderwidth 2 -relief raised -height 50 -width 50
place $Name -x 416 -relx 0 -y 208 -rely 0 -width 144 -relwidth
0 -height 264 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f0.lb1
#----------------------------------------
listbox $Name -font -*-*-*-*-Normal--12-*-*-*-c-*-*-* -height 1
-width 0 -yscrollcommand "$Parent.f0.sb2 set"
pack $Name -anchor center -expand 1 -fill both -ipadx 0 -ipady
0 -padx 5 -pady 5 -side left
placement_store "$Name" "pack"

set Name $Parent.f0.sb2
#----------------------------------------
scrollbar $Name -command "$Parent.f0.lb1 yview"
pack $Name -anchor center -expand 0 -fill y -ipadx 0 -ipady 0 -
padx 5 -pady 5 -side left
placement_store "$Name" "pack"

set Name $Parent.b_teacher
#----------------------------------------
checkbutton $Name -activeforeground #000000 -anchor w -
background #64c8c8 -highlightbackground #64c8c8 -highlightcolor
#000000 -text {Is Teacher} -variable wst1_is_teacher -width 8
place $Name -x 4 -relx 0 -y 104 -rely 0 -width 120 -relwidth 0
-height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .trans_start.b_teacher <ButtonRelease-1> {

global wst1_id_input

  if {$wst1_is_teacher == 0} {
    if {($wst1_id_input == "TEACHER") || ($wst1_id_input == ""
)} {
      set wst1_id_input "student"
    }
  } else {
    if { ($wst1_id_input == "student") || ($wst1_id_input ==
"") } {
      set wst1_id_input "TEACHER"
    }
  }

}

set Name $Parent.f153
#----------------------------------------
frame $Name -borderwidth 2 -relief raised -background #0000ff -
height 50 -highlightbackground #0000ff -width 50
place $Name -x 149 -relx 0 -y 6 -rely 0 -width 90 -relwidth 0 -
height 136 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f153.b155
#----------------------------------------
radiobutton $Name -anchor w -background #0000ff -
highlightbackground #0000ff -text CellB -value CellB -variable
wst1_coding -width 8
place $Name -x 15 -relx 0 -y 35 -rely 0 -width 65 -relwidth 0 -
height 20 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f153.b157
#----------------------------------------
radiobutton $Name -anchor w -background #0000ff -
highlightbackground #0000ff -text Jpeg -value Jpeg -variable
wst1_coding -width 8
place $Name -x 15 -relx 0 -y 55 -rely 0 -width 60 -relwidth 0 -
height 20 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f153.b159
#----------------------------------------
radiobutton $Name -anchor w -background #0000ff -
highlightbackground #0000ff -text UYVY -value UYVY -variable
wst1_coding -width 8
place $Name -x 15 -relx 0 -y 75 -rely 0 -width 65 -relwidth 0 -
height 20 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f153.l161
#----------------------------------------
label $Name -anchor w -background #0000ff -highlightbackground
#0000ff -text {Video Coding}
place $Name -x 5 -relx 0 -y 0 -rely 0 -width 80 -relwidth 0 -
height 25 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
```

```
set Name $Parent.f153.b153
#----------------------------------------
radiobutton $Name -text CellB -value CellB -variable
wst1_coding -width 8
place $Name -x 80 -relx 0 -y 175 -rely 0 -width 74 -relwidth 0
-height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f153.b154
#----------------------------------------
radiobutton $Name -anchor w -background #0000ff -
highlightbackground #0000ff -text Mpeg1 -value Mpeg1 -variable
wst1_coding -width 8
place $Name -x 15 -relx 0 -y 95 -rely 0 -width 65 -relwidth 0 -
height 20 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b121
#----------------------------------------
checkbutton $Name -anchor w -background #64c8c8 -
highlightbackground #64c8c8 -text Debug -variable
wst1_enable_debug -width 8
place $Name -x 120 -relx 0 -y 152 -rely 0 -width 80 -relwidth 0
-height 20 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b126
#----------------------------------------
checkbutton $Name -anchor w -background #64c8c8 -command
w_st1:stm_show -highlightbackground #64c8c8 -text ATM -variable
wst1_use_atm -width 8
place $Name -x 120 -relx 0 -y 240 -rely 0 -width 50 -relwidth 0
-height 25 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.e_atm
#----------------------------------------
entry $Name -background white -highlightbackground #64c8c8 -
textvariable atm_adr
place $Name -x 172 -relx 0 -y 240 -rely 0 -width 69 -relwidth 0
-height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b128
#----------------------------------------
checkbutton $Name -anchor w -background #64c8c8 -
highlightbackground #64c8c8 -text {Video Free} -variable
vid_uncontrol -width 8
place $Name -x 4 -relx 0 -y 124 -rely 0 -width 140 -relwidth 0
-height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b185
#----------------------------------------
checkbutton $Name -anchor w -background #64c8c8 -
highlightbackground #64c8c8 -text {No Multimedia} -variable
no_mm -width 8
place $Name -x 4 -relx 0 -y 144 -rely 0 -width 114 -relwidth 0
-height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b193
#----------------------------------------
checkbutton $Name -anchor w -background #64c9c8 -
highlightbackground #64c8c8 -text {No camera} -variable
wst1_no_camera -width 8
place $Name -x 120 -relx 0 -y 172 -rely 0 -width 92 -relwidth 0
-height 20 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b197
#----------------------------------------
checkbutton $Name -anchor w -background #64c8c8 -
highlightbackground #64c8c8 -text {Half duplex} -variable
wst1_half_dupl -width 8
place $Name -x 120 -relx 0 -y 192 -rely 0 -width 104 -relwidth
0 -height 20 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.e_n
#----------------------------------------
entry $Name -background white -highlightbackground #64c8c8 -
textvariable wst1_id_input
place $Name -x 6 -relx 0 -y 17 -rely 0 -width 137 -relwidth 0 -
height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b201
#----------------------------------------
checkbutton $Name -anchor nw -background #64c8c8 -
highlightbackground #64c8c8 -text {Show pictures} -variable
wst1_pict -width 8
place $Name -x 120 -relx 0 -y 212 -rely 0 -width 116 -relwidth
0 -height 24 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
}
proc mainwincr_floor_info {nr} {
set w [win_name .floor_info $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -background #0000ff
set_onkill $w {whide $w}
wm protocol $w WM_DELETE_WINDOW "whide $w"
if {$nr > 0} {
wm title $w "floor_information_${nr}"
} else {wm title $w "floor_information"}
set Toppos($w) 1
set Topgeom($w) 1
set stretchX($w) 0
set stretchY($w) 1
set Topmin($w) 1
set TopIsModule($w) 0
global is_guibuilder
wm geometry $w 338x123
wm geometry $w +135+404
wm resizable $w 0 1
wm minsize $w 338 123
if {!$is_guibuilder} {
  defaultpos_restore $w 1 0
  defaultpos_prepsave $w
}

set Name $w
set Parent $Name
bind .floor_info <Alt-h> {

  w_fi:hide

}
bind .floor_info <KeyPress-Return> {

  w_fi:hide

}
bind .floor_info <KeyPress-Escape> {

  w_fi:hide

}

set Name $Parent.f
#----------------------------------------
frame $Name -borderwidth 2 -relief raised -background #64c8c8 -
height 50 -width 50
place $Name -x 4 -relx 0 -y 8 -rely 0 -width 326 -relwidth 0 -
height -40 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f.sb178
#----------------------------------------
scrollbar $Name -command "$Parent.f.fb1 yview"
place $Name -x 291 -relx 0 -y 23 -rely 0 -width 20 -relwidth 0
-height -32 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f.fb1
#----------------------------------------
fancylistbox $Name -background gray80 -height 5 -
insertbackground gray80 -width 10 -yscrollcommand
"$Parent.f.sb178 set"
place $Name -x 0 -relx 0 -y 24 -rely 0 -width 296 -relwidth 0 -
height -32 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f.f183
#----------------------------------------
frame $Name -borderwidth 2 -relief sunken -height 50 -width 50
place $Name -x 137 -relx 0 -y 23 -rely 0 -width 4 -relwidth 0 -
height -32 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f.f188
#----------------------------------------
frame $Name -borderwidth 2 -relief sunken -height 50 -width 50
place $Name -x 189 -relx 0 -y 23 -rely 0 -width 4 -relwidth 0 -
height -32 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f.f190
#----------------------------------------
frame $Name -borderwidth 2 -relief sunken -height 50 -width 50
place $Name -x 240 -relx 0 -y 23 -rely 0 -width 4 -relwidth 0 -
height -32 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f.l194
#----------------------------------------
label $Name -anchor w -background #64c8c8 -text {Floor name}
place $Name -x 12 -relx 0 -y 4 -rely 0 -width 63 -relwidth 0 -
height 11 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f.l196
#----------------------------------------
label $Name -anchor w -background #64c8c8 -text Control
place $Name -x 143 -relx 0 -y 6 -rely 0 -width 45 -relwidth 0 -
height 10 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f.l198
#----------------------------------------
label $Name -anchor w -background #64c8c8 -text Read
place $Name -x 246 -relx 0 -y 6 -rely 0 -width 35 -relwidth 0 -
height 10 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f.l200
#----------------------------------------
label $Name -anchor w -background #64c8c8 -text Write
place $Name -x 200 -relx 0 -y 6 -rely 0 -width 35 -relwidth 0 -
height 10 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b182
#----------------------------------------
button $Name -command w_fi:hide -highlightbackground #0000ff -
text Hide -underline 0
```

```
place $Name -x 126 -relx 0 -y -29 -rely 1 -width 120 -relwidth
0 -height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
}
proc mainwincr_start2 {nr} {
set w [win_name .start2 $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w -background #64c8c8
set_onkill $w {sy:quit_all}
wm protocol $w WM_DELETE_WINDOW "sy:quit_all"
if {$nr > 0} {
wm title $w "Start DTC system_${nr}"
} else {wm title $w "Start DTC system"}
set Toppos($w) 1
set Topgeom($w) 1
set stretchX($w) 0
set stretchY($w) 0
set Topmin($w) 1
set TopIsModule($w) 0
global is_guibuilder
wm geometry $w 217x203
wm geometry $w +191+227
wm resizable $w 0 0
wm minsize $w 217 203
if {!$is_guibuilder} {
  defaultpos_restore $w 1 0
  defaultpos_prepsave $w
}

set Name $w
set Parent $Name
bind .start2 <Alt-s> {

  .start2.b_start invoke

}
bind .start2 <Alt-q> {

 .start2.b_quit invoke

}
bind .start2 <KeyPress-Return> {

  .start2.b_start invoke

}
bind .start2 <Alt-i> {

   focus .start2.e_nr

}
bind .start2 <Alt-m> {

  focus .start2.m_inp
  w_st2:show_m_menu

}

set Name $Parent.l178
#-----------------------------------------
label $Name -anchor w -relief sunken -text READY -textvariable
er_message
place $Name -x 5 -relx 0 -y 170 -rely 0 -width 145 -relwidth 0
-height 25 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.l180
#-----------------------------------------
label $Name -anchor w -background #64c8c8 -text Status
place $Name -x 5 -relx 0 -y 150 -rely 0 -width -175 -relwidth 1
-height -185 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.l182
#-----------------------------------------
label $Name -anchor w -background #64c8c8 -text {ID nr.} -
underline 0
place $Name -x 11 -relx 0 -y 23 -rely 0 -width 40 -relwidth 0 -
height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.l184
#-----------------------------------------
label $Name -anchor w -background #64c8c8 -text {Meeting nr.} -
underline 0
place $Name -x 11 -relx 0 -y 51 -rely 0 -width 74 -relwidth 0 -
height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.m_inp
#-----------------------------------------
entry $Name -background white -highlightbackground #64c8c8 -
textvariable wst2_meetingnr_inp
place $Name -x 91 -relx 0 -y 51 -rely 0 -width 91 -relwidth 0 -
height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b_start
#-----------------------------------------
button $Name -command {w_st2:start_conference 0} -
highlightbackground #64c8c8 -text Start -underline 0
place $Name -x 17 -relx 0 -y 91 -rely 0 -width 74 -relwidth 0 -
height 40 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b_quit
#-----------------------------------------
button $Name -command w_st1:quit -highlightbackground #64c8c8 -
text Quit -underline 0
place $Name -x 160 -relx 0 -y 166 -rely 0 -width 51 -relwidth 0
-height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .start2.b_quit <KeyPress-Return> {

  %W invoke

}

set Name $Parent.e_nr
#-----------------------------------------
entry $Name -background white -highlightbackground #64c8c8 -
textvariable wst2_usernr_inp
place $Name -x 91 -relx 0 -y 17 -rely 0 -width 109 -relwidth 0
-height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b_mlist
#-----------------------------------------
button $Name -command w_st2:show_m_menu -highlightbackground
#64c8c8 -text ?
place $Name -x 183 -relx 0 -y 51 -rely 0 -width 17 -relwidth 0
-height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
bind .start2.b_mlist <KeyPress-Return> {

  w% invoke

}
}
proc mainwincr_whiteboard {nr} {
set w [win_name .whiteboard $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w
set_onkill $w {destroy $w}
wm protocol $w WM_DELETE_WINDOW "destroy $w"
if {$nr > 0} {
wm title $w "whiteboard_${nr}"
} else {wm title $w "whiteboard"}
set Toppos($w) 0
set Topgeom($w) 1
set stretchX($w) 1
set stretchY($w) 1
set Topmin($w) 1
set TopIsModule($w) 0
global is_guibuilder
wm geometry $w 430x380
wm resizable $w 1 1
wm minsize $w 430 380
if {!$is_guibuilder} {
  defaultpos_restore $w 1 0
  defaultpos_prepsave $w
}

set Name $w
set Parent $Name

set Name $Parent.fr
#-----------------------------------------
frame $Name -borderwidth 2 -relief raised -background #68c8c8 -
height 50 -width 50
place $Name -x 0 -relx 0 -y 40 -rely 0 -width 0 -relwidth 1 -
height -108 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.fr.can
#-----------------------------------------
canvas $Name -background #ffffff -borderwidth 2 -height 40 -
relief sunken -scrollregion {0c 0c 10c 10c} -width 40 -
xscrollcommand "$Parent.fr.sb28 set" -yscrollcommand
"$Parent.fr.sb26 set"
place $Name -x 0 -relx 0 -y 0 -rely 0 -width -17 -relwidth 1 -
height -17 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.fr.sb26
#-----------------------------------------
scrollbar $Name -command "$Parent.fr.can yview"
place $Name -x -17 -relx 1 -y 0 -rely 0 -width 17 -relwidth 0 -
height -17 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.fr.sb28
#-----------------------------------------
scrollbar $Name -command "$Parent.fr.can xview" -orient
horizontal
place $Name -x -1 -relx 0 -y -17 -rely 1 -width -17 -relwidth 1
-height 17 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_st
#-----------------------------------------
frame $Name -borderwidth 2 -relief raised -background #68c8c8 -
height 50 -width 50
place $Name -x 0 -relx 0 -y -68 -rely 1 -width 0 -relwidth 1 -
height 68 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_st.l19
#-----------------------------------------
label $Name -background #68c8c8 -text Status:
```

```
place $Name -x 0 -relx 0 -y 12 -rely 0 -width 52 -relwidth 0 -
height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_st.l30
#----------------------------------------
label $Name -relief sunken -textvariable wb_status
place $Name -x 48 -relx 0 -y 8 -rely 0 -width -140 -relwidth 1
-height 20 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_st.l22
#----------------------------------------
label $Name -background #68c8c8 -text Mode:
place $Name -x 12 -relx 0 -y 36 -rely 0 -width 40 -relwidth 0 -
height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_st.l27
#----------------------------------------
label $Name -relief sunken -textvariable wb_showmode
place $Name -x 48 -relx 0 -y 36 -rely 0 -width 84 -relwidth 0 -
height 20 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_st.l23
#----------------------------------------
label $Name -background #68c8c8 -text Object:
place $Name -x 144 -relx 0 -y 40 -rely 0 -width 48 -relwidth 0
-height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_st.l25
#----------------------------------------
label $Name -relief sunken -text label20 -textvariable
wb_feedback
place $Name -x 192 -relx 0 -y 36 -rely 0 -width -284 -relwidth
1 -height 20 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_st.b14
#----------------------------------------
button $Name -command wb:hide -text Hide
place $Name -x -74 -relx 1 -y 17 -rely 0 -width 57 -relwidth 0
-height 29 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_bt
#----------------------------------------
frame $Name -borderwidth 2 -relief raised -background #68c8c8 -
height 50 -width 50
place $Name -x 0 -relx 0 -y 0 -rely 0 -width 0 -relwidth 1 -
height 40 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_bt.b11
#----------------------------------------
button $Name -command {wb:setMode $wb_can line} -text line
place $Name -x 6 -relx 0 -y 6 -rely 0 -width 40 -relwidth 0 -
height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_bt.b12
#----------------------------------------
button $Name -command {wb:setMode $wb_can oval} -text Oval
place $Name -x 51 -relx 0 -y 6 -rely 0 -width 40 -relwidth 0 -
height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_bt.b13
#----------------------------------------
button $Name -command {wb:setMode $wb_can box} -text Box
place $Name -x 97 -relx 0 -y 6 -rely 0 -width 40 -relwidth 0 -
height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_bt.b18
#----------------------------------------
button $Name -command {wb:setMode $wb_can text} -text Text
place $Name -x 143 -relx 0 -y 6 -rely 0 -width 40 -relwidth 0 -
height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_bt.b10
#----------------------------------------
button $Name -command {wb:erase_selection $wb_can; wb:setMode
$wb_can erase} -text Erase
place $Name -x 251 -relx 0 -y 6 -rely 0 -width 46 -relwidth 0 -
height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.f_bt.b15
#----------------------------------------
button $Name -command {wb:setMode $wb_can stroke} -text Stroke
place $Name -x 189 -relx 0 -y 6 -rely 0 -width 46 -relwidth 0 -
height 23 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
}
proc mainwincr_w_sle {nr} {
set w [win_name .w_sle $nr]
catch "destroy $w"
#- TOP LEVEL ----------------------------------------
global ptype pinfo
global wnr
set wnr $nr
global Topgeom Toppos stretchX stretchY Topmin TopIsModule
global Menu_string
global auto_path images num_images
toplevel $w
set_onkill $w {destroy $w}
wm protocol $w WM_DELETE_WINDOW "destroy $w"
if {$nr > 0} {
wm title $w "Session Layer Emulation_${nr}"
} else {wm title $w "Session Layer Emulation"}
set Toppos($w) 0
set Topgeom($w) 1
set stretchX($w) 1
set stretchY($w) 1
set Topmin($w) 1
set TopIsModule($w) 0
global is_guibuilder
wm geometry $w 512x220
wm resizable $w 1 1
wm minsize $w 512 220
if {!$is_guibuilder} {
  defaultpos_restore $w 1 0
  defaultpos_prepsave $w
}

set Name $w
set Parent $Name

set Name $Parent.l237
#----------------------------------------
label $Name -anchor w -borderwidth 0 -text IP-number
place $Name -x 368 -relx 0 -y 8 -rely 0 -width 48 -relwidth 0 -
height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.l238
#----------------------------------------
label $Name -anchor w -borderwidth 0 -relief sunken -text Name
place $Name -x 368 -relx 0 -y 48 -rely 0 -width 32 -relwidth 0
-height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.l239
#----------------------------------------
label $Name -anchor w -borderwidth 0 -text Floors
place $Name -x 368 -relx 0 -y 88 -rely 0 -width 32 -relwidth 0
-height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.e240
#----------------------------------------
entry $Name -textvariable w_sle_ipnr
place $Name -x 368 -relx 0 -y 24 -rely 0 -width 128 -relwidth 0
-height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.e241
#----------------------------------------
entry $Name -textvariable w_sle_name
place $Name -x 368 -relx 0 -y 64 -rely 0 -width 128 -relwidth 0
-height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b242
#----------------------------------------
button $Name -command w_sle:deleting -text Deleting
place $Name -x 8 -relx 0 -y 40 -rely 0 -width -464 -relwidth 1
-height -200 -relheight 1 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b243
#----------------------------------------
button $Name -command w_sle:adding -text Adding
place $Name -x 8 -relx 0 -y 8 -rely 0 -width 48 -relwidth 0 -
height 24 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b244
#----------------------------------------
button $Name -command w_sle:FP_Tokenplease_ind -text
FP_Tokenplease_ind
place $Name -x 88 -relx 0 -y 8 -rely 0 -width 112 -relwidth 0 -
height 24 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b245
#----------------------------------------
button $Name -command w_sle:FP_Withdraw_ind -text
FP_Withdraw_ind
place $Name -x 88 -relx 0 -y 40 -rely 0 -width 112 -relwidth 0
-height 24 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b246
#----------------------------------------
button $Name -command w_sle:FC_Tokengrab_ind -text
FC_tokengrab_ind
place $Name -x 224 -relx 0 -y 72 -rely 0 -width 104 -relwidth 0
-height 24 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b247
#----------------------------------------
button $Name -command w_sle:Tokengive_ind -text Tokengive_ind
place $Name -x 224 -relx 0 -y 8 -rely 0 -width 104 -relwidth 0
-height 24 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b248
#----------------------------------------
button $Name -command w_sle:FC_Reject_ind -text FC_Reject_ind
place $Name -x 224 -relx 0 -y 40 -rely 0 -width 104 -relwidth 0
-height 24 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b249
#----------------------------------------
button $Name -command w_sle:FC_Reset_ind -text FC_Reset_ind
place $Name -x 224 -relx 0 -y 104 -rely 0 -width 104 -relwidth
0 -height 24 -relheight 0 -anchor nw -bordermode inside
```

```
placement_store "$Name" "place"

set Name $Parent.e250
#----------------------------------------
entry $Name -textvariable w_sle_floors
place $Name -x 368 -relx 0 -y 104 -rely 0 -width 128 -relwidth
0 -height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.l252
#----------------------------------------
label $Name -anchor w -borderwidth 0 -text {Text / Userdata}
place $Name -x 8 -relx 0 -y 168 -rely 0 -width 80 -relwidth 0 -
height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b253
#----------------------------------------
button $Name -command w_sle:text_stream_ind -text
Text_stream_ind
place $Name -x 8 -relx 0 -y 136 -rely 0 -width 96 -relwidth 0 -
height 24 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.e255
#----------------------------------------
entry $Name -textvariable w_sle_text
place $Name -x 8 -relx 0 -y 184 -rely 0 -width 488 -relwidth 0
-height 24 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.b232
#----------------------------------------
button $Name -command w_sle:FC_Tokengrab_conf -text
FC_Tokengrab_conf
place $Name -x 88 -relx 0 -y 72 -rely 0 -width 112 -relwidth 0
-height 24 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.e233
#----------------------------------------
entry $Name -textvariable w_sle_channel
place $Name -x 368 -relx 0 -y 144 -rely 0 -width 0 -relwidth
0.25 -height 0 -relheight 0.07273 -anchor nw -bordermode inside
placement_store "$Name" "place"

set Name $Parent.l234
#----------------------------------------
label $Name -anchor w -borderwidth 0 -text channel
place $Name -x 368 -relx 0 -y 128 -rely 0 -width 48 -relwidth 0
-height 16 -relheight 0 -anchor nw -bordermode inside
placement_store "$Name" "place"
}
if {$is_guibuilder} {
wincreate .system
wincreate .network_layer
wincreate .users_db
wincreate .floor_req_grant_db
wincreate .floor_control
wincreate .main
wincreate .floor_use
wincreate .floor_def
wincreate .floor_read_db
wincreate .floor_hci
wincreate .main1
wincreate .media1
wincreate .supercontrol
wincreate .text1
wincreate .freq_popup
wincreate .fref_popup
wincreate .groupmgr
wincreate .trans_start
wincreate .floor_info
wincreate .start2
wincreate .whiteboard
wincreate .w_sle
}
if {!$is_guibuilder} {main}
```